Journal of Applied Mathematics

Intelligent Modeling and Verification 2014

Guest Editors: Guiming Luo, Xiaoyu Song, Xiaojing Yang, and Krishnaiyan Thulasiraman

Intelligent Modeling and Verification 2014

Journal of Applied Mathematics

Intelligent Modeling and Verification 2014

Guest Editors: Guiming Luo, Xiaoyu Song, Xiaojing Yang,
and Krishnaiyan Thulasiraman

Editorial Board

Contents

Intelligent Modeling and Verification 2014, Guiming Luo, Xiaoyu Song, Xiaojing Yang, and Krishnaiyan Thulasiraman
Volume 2014, Article ID 632027, 2 pages

Formalization of Function Matrix Theory in HOL, Zhiping Shi, Zhenke Liu, Yong Guan, Shiwei Ye, Jie Zhang, and Hongxing Wei
Volume 2014, Article ID 201214, 10 pages

Reconstruction of Uncertain Historical Evolution of the Polysyllablization of Chinese Lexis, Bing Qiu and Jie Li
Volume 2014, Article ID 123983, 9 pages

Towards Light-Weight Probabilistic Model Checking, Savas Konur
Volume 2014, Article ID 814159, 15 pages

Multigranulations Rough Set Method of Attribute Reduction in Information Systems Based on Evidence Theory, Minlun Yan
Volume 2014, Article ID 857186, 9 pages

Precomputed Clustering for Movie Recommendation System in Real Time, Bo Li, Yibin Liao, and Zheng Qin
Volume 2014, Article ID 742341, 9 pages

Constitutive Relation of Engineering Material Based on SIR Model and HAM, Haoxiang He, Enzhen Han, and Maolin Cong
Volume 2014, Article ID 624863, 13 pages

Formal Analysis of Fairness for Optimistic Multiparty Contract Signing Protocol, Xiaoru Li, Xiaohong Li, Guangquan Xu, Jing Hu, and Zhiyong Feng
Volume 2014, Article ID 983204, 10 pages

Unified Mathematical Framework for Slicing and Symmetry Reduction over Event Structures, Xinyan Gao, Yingcai Ding, Wenbo Liu, Kaidi Zheng, Siyu Huang, Ning Zhou, and Dakui Li
Volume 2014, Article ID 352152, 20 pages

Compositional Abstraction Refinement for Component-Based Systems, Lianyi Zhang, Qingdi Meng, and Kueiming Lo
Volume 2014, Article ID 703098, 12 pages

Groebner Bases Based Verification Solution for SystemVerilog Concurrent Assertions, Ning Zhou, Xinyan Gao, Jinzhao Wu, Jianchao Wei, and Dakui Li
Volume 2014, Article ID 194574, 15 pages

A Nonlinear Multiparameters Temperature Error Modeling and Compensation of POS Applied in Airborne Remote Sensing System, Jianli Li, Wenjian Wang, Feng Jiao, Jiancheng Fang, and Tao Yu
Volume 2014, Article ID 901539, 11 pages

The Application of Pattern Recognition in Electrofacies Analysis, Huan Li, Xiao Yang, and Wenhong Wei
Volume 2014, Article ID 640406, 8 pages

Formal Modeling and Analysis of Fairness Characterization of E-Commerce Protocols,
Chengwei Zhang, Xiaohong Li, Jing Hu, Zhiyong Feng, and Jiaojiao Song
Volume 2014, Article ID 138370, 10 pages

A Software Vulnerability Rating Approach Based on the Vulnerability Database, Jian Luo, Kueiming Lo, and Haoran Qu
Volume 2014, Article ID 932397, 9 pages

A New Subband Adaptive Filtering Algorithm for Sparse System Identification with Impulsive Noise,
Young-Seok Choi
Volume 2014, Article ID 704231, 7 pages

Robustness Analysis of Floating-Point Programs by Self-Composition, Liqian Chen, Jiahong Jiang, Banghu Yin, Wei Dong, and Ji Wang
Volume 2014, Article ID 789213, 12 pages

Terminal Satisfiability in GSTE, Yongsheng Xu, Guowu Yang, Zhengwei Chang, Desheng Zheng, and Wensheng Guo
Volume 2014, Article ID 725275, 10 pages

Counterexample-Preserving Reduction for Symbolic Model Checking, Wanwei Liu, Rui Wang, Xianjin Fu, Ji Wang, Wei Dong, and Xiaoguang Mao
Volume 2014, Article ID 702165, 13 pages

Approximate Equivalence of the Hybrid Automata with Taylor Theory, Anping He, Jinzhao Wu, Shihan Yang, and Hongyan Tan
Volume 2014, Article ID 850789, 5 pages

Multitask Oriented Virtual Resource Integration and Optimal Scheduling in Cloud Manufacturing,
Zhen Cheng, Dechen Zhan, Xibin Zhao, and Hai Wan
Volume 2014, Article ID 369350, 9 pages

A Case Study on Formal Analysis of an Automated Guided Vehicle System, Jie Zhang, Yuntao Peng, William N. N. Hung, Xiaojuan Li, Jindong Tan, and Zhiping Shi
Volume 2014, Article ID 327465, 10 pages

Modeling, Design, and Implementation of a Cloud Workflow Engine Based on Aneka, Jiantao Zhou, Chaoxin Sun, Weina Fu, Jing Liu, Lei Jia, and Hongyan Tan
Volume 2014, Article ID 512476, 9 pages

Optimal Spatial Matrix Filter Design for Array Signal Preprocessing, Haiyong Zhang, Dong Han, Kai Ji, Zhong Ren, Chi Xu, Lijuan Zhu, and Hongyan Tan
Volume 2014, Article ID 564976, 8 pages

Semantic Consistency Checking in Building Ontology from Heterogeneous Sources, Shihan Yang, Hongyan Tan, and Jinzhao Wu
Volume 2014, Article ID 181938, 11 pages

Test Purpose Oriented I/O Conformance Test Selection with Colored Petri Nets, Jing Liu, Xinming Ye, and Jiantao Zhou
Volume 2014, Article ID 645235, 10 pages

A Bio-Inspired QoS-Oriented Handover Model in Heterogeneous Wireless Networks, Daxin Tian, Jianshan Zhou, Honggang Qi, Yingrong Lu, Yunpeng Wang, Jian Wang, and Anping He
Volume 2014, Article ID 920123, 13 pages

Functional Verification of High Performance Adders in COQ, Qian Wang, Xiaoyu Song, Ming Gu, and Jiaguang Sun
Volume 2014, Article ID 197252, 9 pages

Adaptive Fault-Tolerant Routing in 2D Mesh with Cracky Rectangular Model, Yi Yang, Meirun Chen, Hao Li, and Lian Li
Volume 2014, Article ID 592638, 10 pages

Total Variation Based Perceptual Image Quality Assessment Modeling, Yadong Wu, Hongying Zhang, and Ran Duan
Volume 2014, Article ID 294870, 10 pages

Modeling a Heterogeneous Embedded System in Coloured Petri Nets, Huafeng Zhang, Hehua Zhang, Ming Gu, and Jiaguang Sun
Volume 2014, Article ID 943094, 8 pages

An Efficient Multitask Scheduling Model for Wireless Sensor Networks, Hongsheng Yin, Honggang Qi, Jingwen Xu, Xin Huang, and Anping He
Volume 2014, Article ID 969523, 7 pages

Formal Proof of a Machine Closed Theorem in Coq, Hai Wan, Anping He, Zhiyang You, and Xibin Zhao
Volume 2014, Article ID 892832, 9 pages

Hindawi Publishing Corporation
Journal of Applied Mathematics
Volume 2014, Article ID 632027, 2 pages
http://dx.doi.org/10.1155/2014/632027

Editorial

Intelligent Modeling and Verification 2014

Guiming Luo,[1] Xiaoyu Song,[2] Xiaojing Yang,[3] and Krishnaiyan Thulasiraman[4]

[1] *School of Software, Tsinghua University, Beijing 100084, China*
[2] *Maseeh College of Electrical and Computer Engineering, Portland State University, P.O. Box 751, Portland, OR 97207, USA*
[3] *Department of Mathematical Sciences, Tsinghua University, Beijing 100084, China*
[4] *School of Computer Science, University of Oklahoma, 200 Felgar Street, Room 114, Norman, OK 73019, USA*

Correspondence should be addressed to Guiming Luo; gluo@tsinghua.edu.cn

Received 24 September 2014; Accepted 24 September 2014; Published 22 December 2014

After the success of the previous special issue Intelligent Modeling and Verification, which was published last year, we are pleased to announce that the new special issue Intelligent Modeling and Verification 2014 is finally here. Through an extensive peer-review process, 32 papers were selected for publication in the new special issue.

A total of 16 papers describe optimal system modeling. They focus on system modeling, optimal filtering and scheduling, image modeling, pattern recognition, and network modeling.

Based on the total variation model, the paper entitled "*Total variation based perceptual image quality assessment modeling*" by Y. Wu et al. proposes a new way to assess perceptual image quality in spatial domains. A cluster for use in the film industry is constructed in the paper entitled "*Precomputed clustering for movie recommendation system in real time*" (B. Li et al.) by implementing a distance matrix based on machine learning techniques. The paper entitled "*Unified mathematical framework for slicing and symmetry reduction over event structures*" by X. Gao et al. introduces sliced and symmetric quotient reduction models of event structures and presents some corresponding algorithms. Based on the consideration of sharing and real number matrix encoding, the correlation models of virtual resources and the genetic algorithm are presented in "*Multitask oriented virtual resource integration and optimal scheduling in cloud manufacturing*" (Z. Cheng et al.). The paper entitled "*Optimal spatial matrix filter design for array signal preprocessing*" by H. Zhang et al. illustrates a technique for designing a spatial matrix filter based on convex programming. In the paper entitled "*Constitutive relation of engineering material based on SIR model and HAM*" by H. He et al., a dynamic stress-strain model is established according to the homotopy analysis method, and an analytical solution is presented. The paper entitled "*The application of pattern recognition in electrofacies analysis*" by H. Li et al. introduces the KMRIC algorithm and AKM algorithm. Combining the multigranulations rough set with evidence theory, M. Yan constructs the multigranulations method of the belief and plausibility reductions in the paper entitled "*Multigranulations rough set method of attribute reduction in information systems based on evidence theory*." Complicated data modeling and adaptive filtering are the main areas of focus for the papers "*A new subband adaptive filtering algorithm for sparse system identification with impulsive noise*" (Y.-S. Choi), "*A nonlinear multiparameters temperature error modeling and compensation of POS applied in airborne remote sensing system*" (J. Li et al.), and "*Robustness analysis of floating-point programs by self-composition*" (L. Chen et al.). In order to avoid routing livelock and guarantee multiple terminals' satisfaction, network modeling and scheduling are investigated in "*Adaptive fault-tolerant routing in 2D mesh with cracky rectangular model*" (Y. Yang et al.), "*A bio-inspired QoS-oriented handover model in heterogeneous wireless networks*" (D. Tian et al.), and "*An efficient multitask scheduling model for wireless sensor networks*" (H. Yin et al.). After analyzing the frequencies of CVSS metrics, a vulnerability rating approach is proposed in the paper entitled "*A software vulnerability rating approach based on the vulnerability database*" (J. Luo et al.). Based on the comprehensive dictionary of Chinese words, "*Reconstruction of uncertain historical evolution of the polysyllablization of Chinese lexis*" (B. Qiu and J. Li) presents a new mapping approach from words to their occurrence times.

The remaining 16 papers focus on the fields of formal modeling, checking, and verification.

A hybrid I/O automata model for an automated guided vehicle system is introduced in the paper entitled "*A case study on formal analysis of an automated guided vehicle system*" (J. Zhang et al.). The hybrid automaton and Taylor approximation are studied in the paper entitled "*Approximate equivalence of the hybrid automata with Taylor theory*" (A. He et al.). Based on middle-model methodology, "*Semantic consistency checking in building ontology from heterogeneous sources*" (S. Yang et al.) proposes a model checking method to handle semantic consistency. Using the HOL theorem prover, the higher-order logic formalization of the function vector and the function matrix theories are proposed in the paper entitled "*Formalization of function matrix theory in HOL*" (Z. Shi et al.). Combining the extended modeling language CSP# and linear temporal logic LTL, "*Formal analysis of fairness for optimistic multiparty contract signing protocol*" (by X. Li et al.) verifies the fairness of OMPCS protocols. Embedded system modeling and verification using Petri Nets are explored in the following papers: "*Modeling, design, and implementation of a cloud workflow engine based on Aneka*" (J. Zhou et al.), "*Test purpose oriented I/O conformance test selection with colored Petri Nets*" (J. Liu et al.), and "*Modeling a heterogeneous embedded system in coloured petri Nets*" (H. Zhang et al.). Based on the strand space model, "*Formal modeling and analysis of fairness characterization of E-commerce protocols*" (C. Zhang et al.) proposes a verification method for fairness performance. An approach exploiting the power of polynomial ring algebra is introduced in the paper entitled "*Groebner bases based verification solution for SystemVerilog concurrent assertions*" (N. Zhou et al.) to perform SystemVerilog assertion verification over digital circuit systems. The paper entitled "*Functional verification of high performance adders in Coq*" (Q. Wang et al.) proposes a systematic method to formalize and verify adders in formal proof assistant Coq. A technique for reducing the LTL formula is proposed in "*Counterexample-preserving reduction for symbolic model checking*" (W. Liu et al.). In the paper "*Compositional abstraction refinement for component-based systems*" (L. Zhang et al.), the invariant strengthening and state partitioning techniques help to strengthen the abstraction and find counterexamples. The paper entitled "*Terminal satisfiability in GSTE*" (Y. Xu et al.) gives a six-tuple definition and presents a new algorithm for model checking terminal satisfiability. The paper entitled "*Towards light-weight probabilistic model checking*" (S. Konur) presents a methodology to facilitate probabilistic model checking for nonexperts. A machine closed theorem proof of TLA+ in the theorem proving system Coq is presented in "*Formal proof of a machine closed theorem in Coq*" (H. Wan et al.).

Acknowledgments

We sincerely thank the researchers who contributed to this special issue for their inspiring work, as well as the anonymous reviewers for their diligent assistance. In addition, we would like to express our appreciation for the editorial board members of this journal, who provided valuable help and support throughout the preparation of this special issue.

Guiming Luo
Xiaoyu Song
Xiaojing Yang
Krishnaiyan Thulasiraman

Hindawi Publishing Corporation
Journal of Applied Mathematics
Volume 2014, Article ID 201214, 10 pages
http://dx.doi.org/10.1155/2014/201214

Research Article

Formalization of Function Matrix Theory in HOL

Zhiping Shi,[1,2] Zhenke Liu,[1] Yong Guan,[1] Shiwei Ye,[3] Jie Zhang,[4] and Hongxing Wei[5]

[1] *Beijing Key Laboratory of Electronic System Reliability Technology, Capital Normal University, Beijing 100048, China*
[2] *Guangxi Key Laboratory of Trusted Software, Guilin University of Electronic Technology, Guilin 541004, China*
[3] *College of Information Science and Engineering, Graduate University of Chinese Academy of Sciences, Beijing 100049, China*
[4] *College of Information Science and Technology, Beijing University of Chemical Technology, Beijing 100029, China*
[5] *School of Mechanical Engineering and Automation, Beijing University of Aeronautics and Astronautics, Beijing 100191, China*

Correspondence should be addressed to Zhiping Shi; shizhiping@gmail.com

Received 12 January 2014; Accepted 22 April 2014; Published 24 July 2014

Academic Editor: Guiming Luo

Function matrices, in which elements are functions rather than numbers, are widely used in model analysis of dynamic systems such as control systems and robotics. In safety-critical applications, the dynamic systems are required to be analyzed formally and accurately to ensure their correctness and safeness. Higher-order logic (HOL) theorem proving is a promise technique to match the requirement. This paper proposes a higher-order logic formalization of the function vector and the function matrix theories using the HOL theorem prover, including data types, operations, and their properties, and further presents formalization of the differential and integral of function vectors and function matrices. The formalization is implemented as a library in the HOL system. A case study, a formal analysis of differential of quadratic functions, is presented to show the usefulness of the proposed formalization.

1. Introduction

Being operators of linear space transformation, matrices have extended their applications in many science fields such as physics, mechanics, optics, the probability theory, and many engineering fields such as computer graphics, signal processing, and robotics. In the applications, dynamic system modeling requests function matrices, in which elements are functions rather than constants.

Traditionally, function matrix computations are dealt with by numerical analysis algorithms or computer algebra algorithms, yet the absolute precision in the real number field can never be reached because of round-off error, approximate algorithms to address large-scale issues, and so on. On the other hand, analysis of function matrix based models has been carried out with paper and pencil, as is quite tedious and error-prone. A tiny error or inaccuracy, however, may result in failure or even loss of lives in highly sensitive and safety-critical engineering applications. Mechanical theorem proving, on the contrary, is capable of performing precise and scalable analysis.

Mechanical theorem proving has been considered a promising and powerful method of formal proofs in pure mathematics or system analysis and verification [1–5]. Systems or any proof goals need to be modeled formally before they are verified by theorem provers, and theorem provers work based on logic theorem libraries of mathematics. It is indispensable to formalize function matrix theory before formal verifying the systems based on the theory. Real matrices have been formalized in many theorem provers. Nakamura et al. [6] presented the formalization of the matrix theory in Mizar in 2006. The COQ system initiated to provide matrices in recent years [7]. Harrison presented the formalization of Euclidean space in the HOL Light system in 2005 [8]. In Isabelle/HOL [9], some basic matrix theory has been formalized [10, 11]. We have developed the basic matrix theory in HOL theorem prover [12]. However, no formalized function matrix theory has yet been reported in literatures.

HOL is one of the most popular theorem provers and has a lot of successful applications. The newest version of the HOL is named HOL4. This paper introduces the formalization of the function matrix theory in HOL4, including formalization of definitions and properties of function vectors and function matrices as well as their arithmetic operations and differential. This work is jointly based on the realTheory library, limTheory library, and fcpTheory library in HOL4.

The formalization of the function vector is proposed in Section 2, while the formalization of the function matrix occupied Section 3. Section 4 proposes formal definitions and properties of differential and integral of function matrices (vectors). As a case study, formal proof of differential of quadratic functions is shown in Section 5 to demonstrate the usefulness of the formalized theory. Finally, the paper draws the conclusion in Section 6.

2. Formalization of Function Vectors

Formal definitions and properties of function vectors and their operations are proposed in this section.

A vector with functions of a variable x as elements is called a function vector, denoted by

$$\alpha(x) = (\alpha_1(x), \ldots, \alpha_i(x), \ldots, \alpha_n(x)). \quad (1)$$

The difference between a function vector and a real vector is that the elements of the function vector are functions although a real number could also be seen as a constant function. In HOL4, "α -> β" donates a function type with domain α and range β. In this paper the elements of a function vector are real functions, and the data type of functions is denoted by "real -> real."

In HOL4, the library fcpTheory provides an operator "$**$" to construct multidimensional data types. So, the function vector data type is defined by

```
Hol_type: (real -> real) * * 'n,
```

where "'n" is a type variable denoting the dimension of function vectors, and the actual dimension can be retrieved by the function dimindex (:'n).

According to the data type definition above, for a function vector **v** of this type, the element v_i at position i is denoted by "v %% i" or "v ' i" in HOL4. Based on the definition, the arithmetic operations and their properties of function vectors are formalized below.

Based on the type definition, we present the formal definitions of the arithmetic operations of function vectors and the formal definitions of some special function vectors, such as the base vectors and the zero vectors in HOL4. In this paper, **fv**, **fv**1, and **fv**2 denote function vectors, f and g real functions, and k and l real numbers. To simplify the definitions of arithmetic operations, two mapping functions are given in Definitions 1 and 2, which expose their elements to operations of function vectors.

Definition 1 (fvector_map_def). |- !f fv. fvector_map f fv = FCP i. (\x. f (fv ' i x)).

Definition 2 (fvector_map2_def). |- !f fv1 fv2. fvector_map2 f fv1 fv2 = FCP i. (\x. f (fv1 ' i x) (fv2 ' i x)).

Definition 1 is a unary operation and Definition 2 is a binary operation on function vectors. Based on the two definitions, the arithmetic operations of function vectors could be defined concisely. "~", "+," and "-" are overloaded for negative, addition, and subtraction of function vectors.

Definition 3 (fvector_neg_def). |- $ ~ = fvector_map numeric_negate.

Definition 4 (fvector_add_def). |- $ + = fvector_map2 $+.

Definition 5 (fvector_sub_def). |- $ - = fvector_map2 $-.

Two function vectors can do inner product operation, calculated as the following formula:

$$v(x)\,u(x) = \sum_{i=0}^{n} v_i(x)\,u_i(x). \quad (2)$$

In HOL4, "$**$" is used to represent the inner product operator.

Definition 6 (fvector_dot_def). |- !fv1 fv2. fv1 ** fv2 = (\x. sum (0, dimindex (:'n)) (\i. fv1 ' i x * fv2 ' i x)).

A function vector can be multiplied by a scalar. Here, we formalize the operations multiplying function vectors by real numbers and real functions on left and on right, respectively.

Definition 7 (fvector_mul_lk_def). |- !k fv. k ** fv = FCP i. (\x. k * fv ' i x).

Definition 8 (fvector_mul_rk_def). |- !fv k. fv ** k = FCP i. (\x. fv ' i x * k).

Definition 9 (fvector_mul_lkx_def). |- !kx fv. kx ** fv = FCP i. (\x. kx x * fv ' i x).

Definition 10 (fvector_mul_rkx_def).

|- !fv kx. fv ** kx = FCP i.(\x. fv ' i x * kx x),

where kx denotes a real function contrasting k for a real number.

We present the definitions of the zero vectors and the basis vectors, and the elements of these function vectors are the constant functions 0 and 1, respectively.

Definition 11 (fvector_0_def). |- fvector_0 = FCP i. (\x. 0).

Definition 12 (fvector_basis_def). |- !k. fvector_basis k = FCP i. if i = k then (\x. 1) else (\x. 0).

It is useful to compute the value of a function vector at a certain x, as is defined by Definition 13.

Definition 13 (compute_fvector_def). |- !fv x. compute_fvector fv x = FCP i. fv ' i x.

On the basis of the definitions formalized above, we formalize a number of the operations' properties and all the properties are proven to be HOL4 theorems. Most of the properties are of linearity and direct-viewing. We list parts of the properties in Table 1.

Table 1: Part operations' properties of function vectors.

Property name	Formalization
FVECTOR_NEG	|- !fv. ~fv = −1 ** fv
FVECTOR_ADD_MUL_LK	|- !fv1 fv2 k. k ** (fv1 + fv2) = k ** fv1 + k ** fv2
FVECTOR_ADD_RDISTRIB	|- !fv1 fv2 fv3. (fv1 + fv2) ** fv3 = (\x. (fv1 ** fv3) x + (fv2 ** fv3) x)
FVECTOR_ADD_ASSOC	|- !fv1 fv2 fv3. fv1 + fv2 + fv3 = fv1 + (fv2 + fv3)
FVECTOR_SUB_LZERO	|- !fv. fvector_0 − fv = ~fv
FVECTOR_DOT_FBASIS	|- !fv k x. k < dimindex (:'n) ==> (fv ** fvector_basis k = fv ' k)
FVECTOR_DOT_FCP	|- ($FCP fv1 ** fv2 = (\x. sum (0,dimindex (:'n)) (\i. fv1 i x * fv2 ' i x))) ∧ (fv2 ** $FCP fv1 = (\x. sum (0,dimindex (:'n)) (\i. fv2 ' i x * fv1 i x)))
FVECTOR_ADD_INDEX	|- !fv1 fv2 i. i < dimindex (:'n) ==> ((fv1 + fv2) ' i = (\x. fv1 ' i x + fv2 ' i x))
FVECTOR_SUB_INDEX	|- !fv1 fv2 i. i < dimindex (:'n) ==> ((fv1 − fv2) ' i = (\x. fv1 ' i x − fv2 ' i x))
FVECTOR_NEG_NEG	|- !fv. ~~fv = fv
FVECTOR_ADD_MUL_LKX	|- !fv1 fv2 kx. kx ** (fv1 + fv2) = kx ** fv1 + kx ** fv2
FVECTOR_ADD_MUL_RKX	|- !fv1 fv2 kx. (fv1 + fv2) ** kx = fv1 ** kx + fv2 ** kx
FVECTOR_MUL_LRADD	|- !fv k l. (k + l) ** fv = k ** fv + l ** fv
FVECTOR_MUL_RRADD	|- !fv k l. fv ** (k + l) = fv ** k + fv ** l
FVECTOR_MUL_LFADD	|- !fv f g. (\x. f x + g x) ** fv = f ** fv + g ** fv
FVECTOR_ADD_RDISTRIB	|- !fv1 fv2 fv3. (fv1 + fv2) ** fv3 = (\x. (fv1 ** fv3) x + (fv2 ** fv3) x)
FVECTOR_SUB_LDISTRIB	|- !fv1 fv2 fv3. fv1 ** (fv2 − fv3) = (\x. (fv1 ** fv2) x − (fv1 ** fv3) x)
FVECTOR_SUB_RDISTRIB	|- !fv1 fv2 fv3. (fv1 − fv2) ** fv3 = (\x. (fv1 ** fv3) x − (fv2 ** fv3) x)
FVECTOR_DOT_LMUL_K	|- !fv1 fv2 k. (\x. k * (fv1 ** fv2) x) = (k ** fv1) ** fv2
FVECTOR_DOT_LMUL_KX	|- !fv1 fv2 k. (\x. k x * (fv1 ** fv2) x) = (k ** fv1) ** fv2
FVECTOR_MUL_LK_ASSOC	|- !fv k l. k ** l ** fv = (k * l) ** fv
FVECTOR_MUL_LKX_ASSOC	|- !fv f g. f ** g ** fv = (\x. f x * g x) ** fv
FVECTOR_DOT_COMM	|- !fv1 fv2. fv1 ** fv2 = fv2 ** fv1
FVECTOR_EQ	|- !fv1 fv2. (fv1 = fv2) <=> (fv1 − fv2 = fvector_0)
FVECTOR_EQ2	|- !fv1 fv2. (fv1 = fv2) <=> !i. i < dimindex (:'n) ==> (fv1 ' i = fv2 ' i)
FVECTOR_ADD_LID	|- !fv. fvector_0 + fv = fv
FVECTOR_ADD_RID	|- !fv. fv + fvector_0 = fv
FVECTOR_ADD_NEG	|- !fv. fv + ~fv = fvector_0
FVECTOR_ADD_NEG2	|- !fv1 fv2. fv1 + ~fv2 = fv1 − fv2
FVECTOR_SUB_ADD	|- !fv1 fv2. fv1 − fv2 + fv2 = fv1
FVECTOR_MUL_L1	|- !fv. 1 ** fv = fv
FVECTOR_LNEG_UNIQ	|- !fv1 fv2. (fv1 + fv2 = fvector_0) <=> (fv1 = ~fv2)
FVECTOR_RNEG_UNIQ	|- !fv1 fv2. (fv1 + fv2 = fvector_0) <=> (fv2 = ~fv1)
FVECTOR_MULK_COMM	|- !fv k. fv ** k = k ** fv
FVECTOR_MULKX_COMM	|- !fv f. fv ** f = f ** fv
FVECTOR_EXIST_NEG	|- !fv. ?fv'. fv + fv' = fvector_0
FVECTOR_FVECTOR_0_DOT	|- !fv. fvector_0 ** fv = (\x. 0)
COMPUTE_FVEC_MUL_MATRIX	|- !fv A x. compute_fvector fv x ** A = compute_fvector (fv ** A) x
COMPUTE_VEC_MUL_FVEC	|- !fv v x. v ** compute_fvector fv x = (v ** fv) x

3. Function Matrices

Like defining function vectors, "**" is used again to define function matrices. A function matrix takes function vectors with data type "(real -> real) * * 'n" as elements. So, the data type of function matrices is formally defined using "* *" twice as

Hol_type: ((real -> real) * * 'n) * * 'm.

This defines a function matrix with dimindex (:'n) rows and dimindex (:'m) columns. Similar to function vectors, "A %% i %% j" or "A ' i ' j" refers to the element of the *i*th row and *j*th column of the function matrix *A*.

Based on the type definition, we present formal definitions of the arithmetic operations of function matrices, including negative, addition, subtraction, transposition and multiplication by function matrices, function vectors, and scalars and functions. And the formal definitions of the special function matrices, the identity matrices, and the zero matrixes are presented. In addition, the function matrices' inversion is formally defined. In this paper, fm, fm1, and fm2 symbolize function matrices, f and g functions, and k and l real numbers. Two mapping functions are defined to make formalizations of negative, addition, and subtraction concise, like in the function vectors case.

Definition 14 (fmatrix_map_def). |- !f fm. fmatrix_map f fm = FCP i j. (\x. f (fm ' i ' j x)).

Definition 15 (fmatrix_map2_def). |- !f fm1 fm2. fmatrix_map2 f fm1 fm2 = FCP i j. (\x. f (fm1 ' i ' j x)(fm2 ' i ' j x)).

Definition 16 (fmatrix_neg_def). |- fmatrix_neg = fmatrix_map numeric_negate.

Definition 17 (fmatrix_add_def). |- $+ = fmatrix_map2 $+.

Definition 18 (fmatrix_sub_def). |- $- = fmatrix_map2 $-.

Multiplication of function matrices is based on the multiplication of rows and columns of the function matrices. The functions to retrieve a certain row or column of a function matrix are formalized based on FCP.

Definition 19 (fun_row_def). |- !fm k. fun_row fm k = FCP j. fm ' k ' j.

Definition 20 (fun_column_def). |- !fm k. fun_column fm k = FCP i. fm ' i ' k.

Definition 21 (fmatrix_prod_def). |- !fm1 fm2. fm1 ** fm2 = FCP i j. fun_row fm1 i ** fun_column fm2 j.

The function matrices can be multiplied with different data types including function factors, real numbers, and real functions. The formal definitions are as follows.

Definition 22 (fmatrix_mul_lk_def). |- !k fm. k ** fm = FCP i j. (\x. k * fm ' i ' j x).

Definition 23 (fmatrix_mul_rk_def). |- !fm k. fm ** k = FCP i j. (\x. fm ' i ' j x * k).

Definition 24 (fmatrix_mul_lkx_def). |- !k fm. k ** fm = FCP i j. (\x. k x * fm ' i ' j x).

Definition 25 (fmatrix_mul_rkx_def). |- !fm k. fm ** k = FCP i j. (\x. fm 'i ' j x * k x).

Definition 26 (fvector_fmatrix_prod_def). |- !fv fm. fv ** fm = FCP i. fv ** fun_column fm i.

Definition 27 (fmatrix_fvector_prod_def). |- !fm fv. fm ** fv = FCP i. fun_row fm i ** fv.

As shown below, we present definitions of the identity function matrix, the zero function matrix, transposed matrices, and reversibility of function matrices.

Definition 28 (fmatrix_0_def). |- fmatrix_0 = FCP i j. (\x. 0).

Definition 29 (fmatrix_E_def). |- fmatrix_E = FCP ij. if i = j then (\x. 1) else (\x. 0).

Definition 30 (transp_fmatrix_def). |- !fm. transp_fmatrix fm = FCP i j. fm ' j ' i.

Definition 31 (fmatrix_inv_def). |- !fm. fmatrix_inv fm <=> ?fm'. (fm ** fm' = fmatrix_E) ∧ (fm' ** fm = fmatrix_E).

Computing values of function matrices on a certain x produces real matrices, as is defined by Definition 32.

Definition 32 (compute_fmatrix_def). |- !fm x. compute_fmatrix fm x = FCP i j. fm ' i ' j x.

Function vectors and function matrices can operate with real vectors and real matrices, which are special function vectors and function matrices.

Based on the definitions above, we formalize many linear properties, which are useful in proving new theorems.

Property 1 (TRANSP_FMATRIX_FCP). The relation between the elements of a function matrix and its transpose is as follows:

|- !fm. transp_fmatrix (FCP i j. fm i j) = FCP i j. fm j i.

Property 2 (TRANSP_TRANSP_FMATRIX). Transposing a function matrix twice changes nothing:

|- !fm. transp_fmatrix (transp_fmatrix fm) = fm.

Property 3 (FMATRIX_PROD_FVECTOR). For any function matrix and function vector, denoted by $A(x)$ and $v(x)$, respectively, it is held that

$$A(x)\,v(x) = v(x)\,A^{T}(x) \tag{3}$$

|- !fm fv. fm ** fv = fv ** transp_fmatrix fm.

Property 4 (FVECTOR_PROD_FMATRIX). Swapping the positions of the function matrix and the function vector, it is held that

$$v(x)\,A(x) = A^{T}(x)\,v(x) \tag{4}$$

|- !fm fv. fv ** fm = transp_fmatrix fm ** fv.

Property 5 (TRANSP_FMATRIX_PROD). For any function matrix $A(x)$, it is held that

$$\left[A^{T}(x)\,A(x)\right]^{T} = A^{T}(x)\,A(x) \tag{5}$$

|- !fm. transp_fmatrix (transp_fmatrix fm ** fm) = transp_fmatrix fm ** fm.

Property 6 (TRANSP_FMATRIX_COLUMN). The rows of a function matrix equal the corresponding columns of its transpose:

```
|- !fm i.i < dimindex (:'m) ==> (fun_column (transp_
fmatrix fm) i = fun_row fm i).
```

Property 7 (TRANSP_FMATRIX_ROW). The columns of a function matrix equal the corresponding rows of its transpose:

```
|- !fm i.i < dimindex (:'n) ==> (fun_row (transp_fmatrix
fm) i = fun_column fm i).
```

Now, we present a special property (Property 8). A function matrix, denoted by $A(x)$, can be formed by column function vectors, written as

$$A(x) = [v_1(x), \ldots, v_i(x), \ldots, v_n(x)]. \tag{6}$$

So, multiplying a function matrix by its transpose can be calculated as

$$A^T(x) A(x) = [v_i(x) v_j(x)]_{n\times n}. \tag{7}$$

The property is formalized in Property 8.

Property 8 (FMATRIX_ROW_PROD). |- !fm. transp_fmatrix fm ∗∗ fm = FCP i j. fun_column fm i ∗∗ fun_column fm j.

Property 9 (FMATRIX_VECTOR_DOT_PROD_FMATRIX). For multiplication of a function matrix and a function vector, it is held that

$$A(x) v(x) A(x) = v(x) A^T(x) A(x) \tag{8}$$

```
|- !fm v. (fm ∗∗ v) ∗∗ fm = v ∗∗ transp_fmatrix fm
∗∗fm.
```

Property 9 could be derived by Property 3.

Property 10 (COMPUTE_FVEC_MUL_MATRIX). A function vector multiplies a real matrix:

```
|- !fv A x. compute_fvector fv x ∗∗ A = compute_
fvector (fv ∗∗ A) x.
```

Property 11 (COMPUTE_VEC_MUL_FVEC). A real vector multiplies a function vector

```
|- !fv v x. v ∗∗ compute_fvector fv x = (v ∗∗ fv) x.
```

Other properties are listed in Table 2.

In practice, function matrices (and vectors) often operate with real matrices (and vectors), and formalizations of the operations are presented as follows.

Definition 33 (vec_mul_fvector_def). |- !v fv. v ∗∗ fv = (\x. sum (0,dimindex (:'n)) (\i. v 'i ∗ fv 'i x)).

Definition 34 (fvector_mul_vec_def). |- !fv v. fv ∗∗ v = (\x. sum (0,dimindex (:'n)) (\i. fv 'i x ∗ v 'i)).

Definition 35 (vec_mul_fmatrix_def). |- !v fm. v ∗∗ fm = FCP i. v ∗∗ fun_column fm i.

Definition 36 (fmatrix_mul_vec_def). |- !fm v. fm ∗∗ v = FCP i. fun_row fm i ∗∗ v.

Definition 37 (matrix_mul_fvector_def). |- !A fv. A ∗∗ fv = FCP i. row A i ∗∗ fv.

Definition 38 (fvector_mul_matrix_def). |- !fv A. fv ∗∗ A = FCP i. fv ∗∗ column A i.

Definition 39 (matrix_mul_fmatrix_def). |- !A fm. A ∗∗ fm = FCP i j. row A i ∗∗ fun_column fm j.

Definition 40 (fmatrix_mul_matrix_def). |- !fm A. fm ∗∗ A = FCP i j. fun_row fm i ∗∗ column A j.

4. Formalization of Differential and Integral of Function Matrices

A function vector or function matrix is differentiable or integrable supposing all its elements are differentiable or integrable. This section presents the formalizations of differential and integral of function vectors and function matrices based on that of real functions.

A function vector, denoted by $v(x) = (a_i(x))_n$, is derivable at $x = x_0$ if all its elements $a_i(x)$ $(i = 1, 2, \ldots, n)$ are derivable at $x = x_0$, and the derivative can be written as

$$\begin{aligned} v'(x) = \left.\frac{dv(x)}{dx}\right|_{x=x_0} &= \lim_{\Delta x\to 0} \frac{v(x_0+\Delta x) - v(x_0)}{\Delta x} \\ &= \left(a_1'(x) \;\; a_2'(x) \;\; \cdots \;\; a_n'(x)\right). \end{aligned} \tag{9}$$

A function vector, denoted by $v(x) = (a_i(x))_n$, is integrable in $[t_0, t_1]$ if all its elements $a_i(x)$ $(i = 1, 2, \ldots, n)$ are integrable in $[t_0, t_1]$, and the integral can be written as

$$\int_{t_0}^{t_1} v(x)\,dt = \left(\int_{t_0}^{t_1} a_i(x)dt\right)_n. \tag{10}$$

According to the mathematics descriptions, we formally define the differential and integral of function vectors based on that of real functions. The differential and integral of a real function are denoted by "diffl" and "integral" [12], respectively, in HOL4.

Definition 41 (fvector_diffl). For anyone function vector **fv**, it is the case that the differential of **fv** at x is the real vector **v** if and only if the differential of members of **fv** at x equals the corresponding members of **v** at x. In HOL4, it is said that

```
|- !fv v x. (fv fvector_diffl v) x <=> !i. i < dimindex
(:'n) ==> (fv '  i diffl v ' i) x.
```

Definition 42 (fvector_integral). Calculating the integral of a function vector **fv** in $[a, b]$ is equal to calculating the integral of all members of **fv** in $[a, b]$. In HOL4, it is said that

```
|- !a b fv. fvector_integral (a,b) fv = FCP i. integral
(a,b) (fv '  i).
```

Table 2: Properties of operations of function matrices.

Property name	Formalization
COMPUTE_FMATRIX_MUL_EQ	|- !fm1 fm2 x. compute_fmatrix fm1 x ** compute_fmatrix fm2 x = compute_fmatrix (fm1 ** fm2) x
FMATRIX_ADD_INDEX	|- !fm1 fm2 i j. i < dimindex (:'m) ∧ j < dimindex (:'n) ==> ((fm1 + fm2) ' i ' j = (\x. fm1 ' i ' j x + fm2 ' i ' j x))
FMATRIX_SUB_INDEX	|- !fm1 fm2 i j. i < dimindex (:'m) ∧ j < dimindex (:'n) ==> ((fm1 − fm2) ' i ' j = (\x. fm1 ' i ' j x − fm2 ' i ' j x))
FMATRIX_ROW_ADD	|- !fm1 fm2 i. i < dimindex (:'m) ==> (fun_row fm1 i + fun_row fm2 i = fun_row (fm1 + fm2) i)
FMATRIX_COLUMN_ADD	|- !fm1 fm2 i. i < dimindex (:'n) ==> (fun_column fm1 i + fun_column fm2 i = fun_column (fm1 + fm2) i)
FMATRIX_ROW_SUB	|- !fm1 fm2 i. i < dimindex (:'m) ==> (fun_row fm1 i − fun_row fm2 i = fun_row (fm1 − fm2) i)
FMATRIX_COLUMN_SUB	|- !fm1 fm2 i. i < dimindex (:'n) ==> (fun_column fm1 i − fun_column fm2 i = fun_column (fm1 − fm2) i)
FMATRIX_NEG	|- !fm. ~fm = −1 ** fm
FMATRIX_NEG_NEG	|- !fm. ~~fm = fm
FMATRIX_MUL_K_EQ	|- !fm k. fm ** k = k ** fm
FMATRIX_MUL_KX_EQ	|- !fm f. fm ** f = f ** fm
FMATRIX_ADD_COMM	|- !fm1 fm1. fm1 + fm2 = fm2 + fm1
FMATRIX_ADD_ASSOC	|- !fm1 fm2 fm3. fm1 + (fm2 + fm3) = fm1 + fm2 + fm3
FMATRIX_ADD_MUL_LK	|- !fm1 fm2 k. k ** (fm1 + fm2) = k ** fm1 + k ** fm2
FMATRIX_ADD_MUL_RK	|- !fm1 fm2 k. (fm1 + fm2) ** k = fm1 ** k + fm2 ** k
FMATRIX_ADD_MUL_LKX	|- !fm1 fm2 kx. kx ** (fm1 + fm2) = kx ** fm1 + kx ** fm2
FMATRIX_ADD_MUL_RKX	|- !fm1 fm2 kx. (fm1 + fm2) ** kx = fm1 ** kx + fm2 ** kx
FMATRIX_ADD_MUL_LFVEC	|- !fm1 fm2 fv. fv ** (fm1 + fm2) = fv ** fm1 + fv ** fm2
FMATRIX_ADD_MUL_RFVEC	|- !fm1 fm2 fv. (fm1 + fm2) ** fv = fm1 ** fv + fm2 ** fv
FMATRIX_SUB_MUL_LFVEC	|- !fm1 fm2 fv. fv ** (fm1 − fm2) = fv ** fm1 − fv ** fm2
FMATRIX_SUB_MUL_RFVEC	|- !fm1 fm2 fv. (fm1 − fm2) ** fv = fm1 ** fv − fm2 ** fv
FMATRIX_MUL_LRADD	|- !fm k l. (k + l) ** fm = k ** fm + l ** fm
FMATRIX_MUL_RRADD	|- !fm k l. fm ** (k + l) = fm ** k + fm ** l
FMATRIX_MUL_LFADD	|- !fm f g. (\x. f x + g x) ** fm = f ** fm + g ** fm
FMATRIX_MUL_RFADD	|- !fm f g. fm ** (\x. f x + g x) = fm ** f + fm ** g
FMATRIX_MUL_RFVADD	|- !fm fv1 fv2. fm ** (fv1 + fv2) = fm ** fv1 + fm ** fv2
FMATRIX_MUL_LFVADD	|- !fm fv1 fv2. (fv1 + fv2) ** fm = fv1 ** fm + fv2 ** fm
FMATRIX_ADD_LDISTRIB	|- !fm1 fm2 fm3. fm1 ** (fm2 + fm3) = fm1 ** fm2 + fm1 ** fm3
FMATRIX_ADD_RDISTRIB	|- !fm1 fm2 fm3. (fm1 + fm2) ** fm3 = fm1 ** fm3 + fm2 ** fm3
FMATRIX_MUL_LMUL_K	|- !fm1 fm2 k. k ** fm1 ** fm2 = (k ** fm1) ** fm2
FMATRIX_MUL_RMUL_K	|- !fm1 fm2 k. k ** fm1 ** fm2 = fm1 ** k ** fm2
FMATRIX_MUL_NEG	|- !fm1 fm2. ~fm1 ** fm2 = fm1 ** ~fm2
FMATRIX_NEG_PROD	|- !fm1 fm2. ~fm1 ** fm2 = ~(fm1 ** fm2)
FMATRIX_MUL_LMUL_KX	|- !fm1 fm2 kx. kx ** fm1 ** fm2 = (kx ** fm1) ** fm2
FMATRIX_MUL_LK_ASSOC	|- !fm k l. k ** l ** fm = (k * l) ** fm
FMATRIX_MUL_LKX_ASSOC	|- !fm f g. f ** g ** fm = (\x. f x * g x) ** fm
FMATRIX_ADD_LID	|- !fm. fmatrix_0 + fm = fm
FMATRIX_ADD_RID	|- !fm. fm + fmatrix_0 = fm
FMATRIX_ADD_NEG	|- !fm. fm + ~fm = fmatrix_0
FMATRIX_ADD_NEG2	|- !fm1 fm2. fm1 + ~fm2 = fm1 − fm2
FMATRIX_SUB_ADD	|- !fm1 fm2. fm1 − fm2 + fm2 = fm1
FMATRIX_SUB_LZERO	|- !fm. fmatrix_0 − fm = ~fm

Table 2: Continued.

Property name	Formalization
FMATRIX_MUL_L1	|- !fm. 1 ** fm = fm
FMATRIX_MULK_COMM	|- !fm k. fm ** k = k ** fm
FMATRIX_MULKX_COMM	|- !fm kx. fm ** kx = kx ** fm
FVECTOR_PROD_FMATRIX	|- !fm fv. fv ** fm = transp_fmatrix fm ** fv
FMATRIX_FVECTOR_0_PROD	|- !fm. fvector_0 ** fm = fvector_0
FMATRIX_ROW_PROD	|- !fm. transp_fmatrix fm ** fm = FCP i j. fun_column fm i ** fun_column fm j
TRANSP_FMATRIX_COLUMN	|- !fm i. i < dimindex (:'m) ==> (fun_column (transp_fmatrix fm) i = fun_row fm i)
TRANSP_FMATRIX_FVECTOR_PROD	|- !fm fv. fm ** fv = fv ** transp_fmatrix fm
TRANSP_FMATRIX_PROD	|- !fm. transp_fmatrix (transp_fmatrix fm ** fm) = transp_fmatrix fm ** fm
TRANSP_FMATRIX_ROW	|- !fm i. i < dimindex (:'n) ==> (fun_row (transp_fmatrix fm) i = fun_column fm i)

Again, the differentiability and integrability of function vectors are formally defined based on those of real functions. The differentiability and integrability of real functions are denoted by "differentiable" and "integrable" respectively, in HOL4.

Definition 43 (fvector_differentiable). For anyone function vector **fv**, it is the case that **fv** is differentiable at x if and only if all the members of **fv** are differentiable at x. In HOL4, it is said that

```
|- !a b fv. fvector_differentiable fv x <=>

        !a b i. a <= b ∧ i < dimindex (:'n) ==> (fv ' i)
        differentiable x.
```

Definition 44 (fvector_integrable). For anyone function vector **fv**, it is the case that **fv** is integrable in $[a, b]$ if and only if all the members of **fv** are integrable in $[a, b]$. In HOL4, it is said that

```
|- !a b fv. fvector_integrable (a,b) fv <=>

        !a b i. a <= b ∧ i < dimindex (:'n) ==> integrable
        (a,b) (fv ' i).
```

A function matrix, denoted by $A(x) = (a_{ij}(x))_{m*n}$, is derivable at $x = x_0$ if its all elements $a_{ij}(x)$ $(i = 1, 2, \ldots, m;$ $j = 1, 2, \ldots, n)$ are derivable at $x = x_0$, and the derivative can be written as

$$A'(x) = \left.\frac{dA(x)}{dx}\right|_{x=x_0} = \lim_{\Delta x \to 0} \frac{A(x_0 + \Delta x) - A(x_0)}{\Delta x} = \begin{bmatrix} a'_{11}(x_0) & a'_{12}(x_0) & \cdots & a'_{1n}(x_0) \\ a'_{21}(x_0) & a'_{22}(x_0) & \cdots & a'_{2n}(x_0) \\ \cdots & \cdots & \cdots & \cdots \\ a'_{m1}(x_0) & a'_{m2}(x_0) & \cdots & a'_{mn}(x_0) \end{bmatrix}. \quad (11)$$

A function matrix, denoted by $A(x) = (a_{ij}(x))_{m*n}$, is integrable in $[t_0, t_1]$ if all its elements $a_{ij}(x)$ $(i = 1, 2, \ldots, m;$ $j = 1, 2, \ldots, n)$ are integrable in $[t_0, t_1]$, and the integral can be written as

$$\int_{t_0}^{t_1} A(t)\, dt = \left(\int_{t_0}^{t_1} a_{ij}(t) dt\right)_{m\times n}. \quad (12)$$

Similar to function vectors, we formally define the differential and integral of function matrices based on those of real functions as follows.

Definition 45 (fmatrix_diffl). For anyone function matrix fm, it is the case that the differential of fm at x is a matrix A if and only if the differentials of members of fm at x equal the corresponding members of A. In HOL4, it is said that

```
|- !fm A x. (fm fmatrix_diffl A) x <=>

        !i j. i < dimindex (:'m) ∧ j < dimindex (:'n) ==>
        (fm 'i 'j diffl A 'i 'j) x.
```

Definition 46 (fmatrix_integral). Calculating the integral of a function matrix fm in $[a, b]$ is equal to calculating the integral of all members of fm in $[a, b]$. In HOL4, it is said that

```
|- !a b fm. fmatrix_integral (a,b) fm = FCP i j. integral
(a,b) (fm 'i 'j).
```

Definition 47 (fmatrix_differentiable). For anyone function matrix fm, it is the case that fm is differentiable at x if and only if there exists a matrix A which is the differential of fm at x. In HOL4, it is said that

```
|- !fm x. fm fmatrix_differentiable x <=> ?A. (fm
fmatrix_diffl A) x.
```

Definition 48 (fmatrix_ingegrable). For anyone function matrix fm, it is the case that fm is integrable in $[a, b]$ if and only if all the elements of fm are integrable in $[a,b]$. In HOL4, it is said that

```
|- !a b fm. fmatrix_integrable (a,b) fm <=>
        !a b i j. a <= b ∧ i < dimindex (:'m) ∧ j <
        dimindex (:'n) ==>
            integrable (a,b) (fm ' i ' j).
```

Based on the definitions above, we formalize and prove many properties about differential and integral of function matrices. Some of them are presented as follows.

Uniqueness is one of the most important properties for differential. Differential of a function matrix is unique.

Property 12 (FMATRIX_DIFF_UNIQ). |- !fm A B x. (fm fmatrix_diffl A) x ∧ (fm fmatrix_diffl B) x ==> (A = B).

Suppose $A(x) = (a_{ij}(x))_{m*n}$, $B(x) = (b_{ij}(x))_{m*n}$ are differentiable. It is the case that

$$\frac{d}{dx}[A(x) \pm B(x)] = \frac{dA(x)}{dx} \pm \frac{dB(x)}{dx}. \tag{13}$$

The property is formalized in HOL4 as follows.

Property 13 (DIFF_FMATIRX_ADD). |- !fm1 fm2 A B x. (fm1 fmatrix_diffl A) x ∧ (fm2 fmatrix_diffl B) x

```
==>((fm1 + fm2) fmatrix_diffl (A + B)) x.
```

Property 14 (DIFF_FMATIRX_SUB). |- !fm1 fm2 A B x. (fm1 fmatrix_diffl A) x ∧ (fm2 fmatrix_diffl B) x

```
==>((fm1 - fm2) fmatrix_diffl (A - B)) x.
```

Similar to the differential of product of real functions, the differential of inner product of function vectors is defined by

$$\frac{d}{dx}[v_1(x) v_2(x)] = \frac{dv_1(x)}{dx} v_2(x) + v_1(x)\frac{dv_2(x)}{dx}. \tag{14}$$

In HOL4, it is formalized by Property DIFF_FVECTOR_MUL.

Property 15 (DIFF_FVECTOR_MUL). |- !fv1 fv2 V1 V2 x. (fv1 fvector_diffl V1) x ∧ (fv2 fvector_diffl V2)x ==>

```
((fv1 ** fv2) diffl (V1 ** compute_fvector fv2 x +
V2 ** compute_fvector fv1 x)) x.
```

The differential of the product of a function vector and a matrix is defined by

$$\frac{d}{dx}[v(x) A] = \frac{dv(x)}{dx} A. \tag{15}$$

Property 16 (DIFF_FVEC_MUL_MATRIX). |- !A fv v. (fv fvector_diffl v)(x) ==> ((fv ** A) fvector_diffl (v ** A))(x).

Let $k(x)$ be a real function of x, $A(x)$ is a function matrix, and both $k(x)$ and $A(x)$ are differentiable, and then

$$\frac{d}{dx}[k(x) A(x)] = \frac{dk(x)}{dx} A(x) + k(x)\frac{dA(x)}{dx}. \tag{16}$$

Specially, if $k(x)$ regresses to a constant k, then

$$\frac{d}{dx}[kA(x)] = k\frac{dA(x)}{dx}. \tag{17}$$

In HOL4, the above properties are formalized as follows.

Property 17 (DIFF_FMATIRX_MUL_KX). |- !fm A kx k x. (fm fmatrix_diffl A) x ∧ (kx diffl k) x ==>

```
((kx ** fm) fmatrix_diffl (k ** compute_fmatrix
fm x + A ** kx x)) x.
```

Property 18 (DIFF_FMATIRX_MUL_K). |- !fm A k x. (fm fmatrix_diffl A) x ==> ((k ** fm) fmatrix_diffl (k ** A)) x.

Suppose $A(x)$ and $B(x)$ are differentiable, and $A(x)$ and $B(x)$ are multipliable, and then

$$\frac{d}{dx}[A(x) B(x)] = \frac{dA(x)}{dx} B(x) + A(x)\frac{dB(x)}{dx}. \tag{18}$$

Property 19 (DIFF_FMATRIX_MUL). |- !fm1 fm2 A B x. (fm1 fmatrix_diffl A) x ∧ (fm2 fmatrix_diffl B) x ==>

```
((fm1 ** fm2) fmatrix_diffl (compute_fmatrix fm1 x
** B + A ** compute_fmatrix fm2 x)) x.
```

Suppose $A(x)$ is a function matrix, $x = f(t)$ is a real function of t, and $A(x)$ and $f(t)$ are differentiable, and then

$$\frac{d}{dx}A(x) = \frac{dA(x)}{dx} f'(t) = f'(t)\frac{dA(x)}{dx}. \tag{19}$$

Property 20 (DIFF_FMATRIX_CHAIN). |- !fm g A m x. (fm fmatrix_diffl A) (g x) ∧ (g diffl m) x ==>

```
(fmatrix_o fm g fmatrix_diffl (A ** m)) x.
```

That $A(x)$ is a constant matrix is equivalent to that

$$\frac{dA(x)}{dx} = 0. \tag{20}$$

Property 21 (DIFF_CONST_MATRIX). |- !A x. (matrix_to_fun A fmatrix_diffl matrix_0) x.

If $A(x)$ and its inverse are differentiable, then

$$\frac{dA^{-1}(x)}{dx} = -A^{-1}(x)\frac{dA(x)}{dx} A^{-1}(x). \tag{21}$$

Property 22 (FMATRIX_0_INTEGAL). If a function matrix equals the zero function matrix, then the integral of the function matrix is the zero real matrix. In HOL4, it is formalized by

```
|- !fm a b. a <= b ∧ (fm = fmatrix_0) ==> (fmatrix_
integral (a,b) fm = matrix_0).
```

5. Case Study—Differential of Quadratic Functions

For linear control systems, the mathematical models of their performance indicators are quadratic functions of state

```
val DIFF_QUADRATIC = store_thm("DIFF_QUADRATIC",
      "!(fv:'n fun_vector) (v:'n vector) (A:('n,'n) matrix) (t:real).
        (fv fvector_diffl v)(t) ∧ (transp A = A) ==>
        ((fv ** A ** fv) diffl
          (v ** A ** (compute_fvector fv t) + (v ** A) ** (compute_fvector fv t)))(t)",
      REPEAT GEN_TAC THEN
      RW_TAC std_ss [MATRIX_VECTOR] THEN
      '!(fv:'n fun_vector) (A:('n,'n) matrix).
        (transp A = A) ==> (fv ** A ** fv = (fv ** A) ** fv)'
        by REWRITE_TAC [] THENL
      [SRW_TAC [fcpLib.FCP_ss] [fvector_mul_matrix_def] THEN
       SRW_TAC [fcpLib.FCP_ss] [fvector_dot_def] THEN
       ABS_TAC THEN
       MATCH_MP_TAC SUM_EQ THEN
       SRW_TAC [][] THEN
       SRW_TAC [fcpLib.FCP_ss] [fvector_mul_vec_def] THEN
       SRW_TAC [fcpLib.FCP_ss] [matrix_mul_fvec_def] THEN
       SRW_TAC [fcpLib.FCP_ss] [vec_mul_fvector_def] THEN
       GEN_REWR_TAC RAND_CONV [REAL_MUL_COMM] THEN
       REWRITE_TAC [GSYM SUM_CMUL] THEN
       MATCH_MP_TAC SUM_EQ THEN
       SRW_TAC [][] THEN
       DISJ2_TAC THEN
       SRW_TAC [fcpLib.FCP_ss] [row_def, column_def] THEN
       NTAC 3(POP_ASSUM MP_TAC) THEN
       SRW_TAC [fcpLib.FCP_ss] [transp_def] THEN
       PROVE_TAC [REAL_MUL_COMM],ALL_TAC] THEN
      '!(fv:'n fun_vector) (A:('n,'n) matrix) t:real.
         (compute_fvector fv t) ** A = compute_fvector (fv ** A) t'
         by REWRITE_TAC [COMPUTE_FVEC_MUL_MATRIX] THEN
      '!(fv:'n fun_vector) (v:'n vector) t:real.
         v ** (compute_fvector fv t) = (v ** fv) t'
         by REWRITE_TAC [COMPUTE_VEC_MUL_FVEC] THEN
      '!(fv:'n fun_vector) (v:'n vector) (A:('n,'n) matrix) (t:real).
         (fv fvector_diffl v)(t) ==> ((fv ** A) fvector_diffl (v ** A))(t)'
         by REWRITE_TAC [DIFF_FVEC_MUL_MATRIX] THEN
      PROVE_TAC [DIFF_FVECTOR_MUL]);
```

Algorithm 1: Formal proof of the quadratic function differential.

and control variables, and the optimal control problem is called the linear quadratic problem [13]. For example, the differential of quadratic functions is involved in analyzing asymptotic stability of the optimal closed-loop systems. In this section, differential of quadratic functions is formalized.

Let $x = x(t) \in R^n$ be a function vector, and $A = A^T \in R^{n\times n}$ a constant matrix, we formally analyze the differential of the quadratic function $x^T Ax$ with respect to $t_\circ$. Based on the properties of differential of function vectors and matrices, we have

$$\begin{aligned}\frac{d}{dt}\left(x^T Ax\right) &= \frac{dx^T}{dt}Ax + x^T\frac{d}{dt}(Ax)\\ &= \frac{dx^T}{dt}Ax + x^T\left(\frac{dA}{dt}x + A\frac{dx}{dt}\right)\\ &= \frac{dx^T}{dt}Ax + x^T A\frac{dx}{dt}.\end{aligned} \tag{22}$$

The formula is formally proved in HOL4 as shown in Algorithm 1. Following the custom of our formalization, **fv** is employed to denote function vector x and real vector **v** to denote the differential of **fv** at x. $x^T Ax = (x^T A)x$ is proved first to transform the original goal into the differential of the inner product of two function vectors, which has been proven in Property DIFF_FVECTOR_MUL. And the differential of $x^T A$ could be dealt with by Property DIFF_FVEC_MUL_MATRIX.

6. Conclusion

Based on high order logic theorem prover HOL4, this paper formalized the data type definitions and operation definitions of function vectors and function matrices and proved lots of operation properties. This paper also presented the definitions of function matrix differential and integral and their properties. All the formalization was implemented as a library in the HOL4 system. The case study of formal proof of quadratic function illustrated the usefulness of the formalized theory.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

This work was supported by the International Cooperation Program on Science and Technology (2010DFB10930, 2011DFG13000), the National Natural Science Foundation of China (61070049, 61170304, 61104035, 61373034, and 61303014), the Natural Science Foundation of the City of Beijing (4122017), the S&R Key Program of the Beijing Municipal Education Commission (KZ201210028036), and the Open Project Program of State Key Laboratory of Computer architecture and the Open Project Program of Guangxi Key Laboratory trusted software.

References

[1] C. Kern and M. R. Greenstreet, "Formal verification in hardware design: a survey," *ACM Transactions on Design Automation of Electronic Systems*, vol. 4, no. 2, pp. 123–193, 1999.

[2] W. Wu and X. Gao, "Mathematics mechanization and applications after thirty years," *Frontiers of Computer Science in China*, vol. 1, no. 1, pp. 1–8, 2007.

[3] J. Liu and H. Lin, "Proof system for applied Pi calculus," in *Theoretical Computer Science*, vol. 323, pp. 229–243, Springer, Berlin, Germany, 2010.

[4] Y. Li, W. N. N. Hung, and X. Song, "A novel formalization of symbolic trajectory evaluation semantics in Isabelle/HOL," *Theoretical Computer Science*, vol. 412, no. 25, pp. 2746–2765, 2011.

[5] L. Chang, Z. Shi, T. Gu, and L. Zhao, "A family of dynamic description logics for representing and reasoning about actions," *Journal of Automated Reasoning*, vol. 49, no. 1, pp. 1–52, 2010.

[6] Y. Nakamura, N. Tamura, and W. Chang, "A theory of matrices of real elements," *Formalized Mathematics*, vol. 14, no. 1, pp. 21–28, 2006.

[7] I. Pasca, "Formally verified conditions for regularity of interval matrices," in *Intelligent Computer Mathematics*, vol. 6167 of *Lecture Notes in Computer Science*, pp. 219–233, Springer, Berlin, Germany, 2010.

[8] J. Harrison, "A HOL theory of Euclidean space," in *Theorem Proving in Higher Order Logics*, vol. 3603 of *Lecture Notes in Computer Science*, pp. 114–129, Springer, Berlin, Germany, 2005.

[9] T. Nipkow, L. C. Paulson, and M. Wenzel, *Isabelle/HOL: a Proof Assistant for Higher-Order Logic*, vol. 2283 of *Lecture Notes in Computer Science*, Springer, Berlin, Germany, 2002.

[10] S. Obua, *Flyspeck II: the basic linear programs [Ph.D. thesis]*, Technische Universität München, Munich, Germany, 2008.

[11] S. Obua, "Proving bounds for real linear programs in Isabelle/HOL," in *Theorem Proving in Higher Order Logics*, vol. 3603 of *Lecture Notes in Computer Science*, pp. 227–244, Springer, Berlin, Germany, 2005.

[12] Z. Shi, W. Gu, X. Li et al., "The gauge integral theory in HOL4," *Journal of Applied Mathematics*, vol. 2013, Article ID 160875, 7 pages, 2013.

[13] H. U. Shou-song, *Principle of Automatic Control*, Science Press, Beijing, China, 2007.

Hindawi Publishing Corporation
Journal of Applied Mathematics
Volume 2014, Article ID 123983, 9 pages
http://dx.doi.org/10.1155/2014/123983

Research Article

Reconstruction of Uncertain Historical Evolution of the Polysyllablization of Chinese Lexis

Bing Qiu[1] **and Jie Li**[2]

[1] *College of Humanities and Social Sciences, Beijing Language and Culture University, Beijing 100083, China*
[2] *College of Software, Henan University, Kaifeng, Henan 475001, China*

Correspondence should be addressed to Bing Qiu; bingqiu@gmail.com

Received 9 February 2014; Revised 17 June 2014; Accepted 25 June 2014; Published 13 July 2014

Academic Editor: Guiming Luo

Polysyllablization, closely related to phonetics, semantics, and syntactics, is one of the fundamental trends in the development of Chinese lexis. However, with lots of uncertainties in the historical evolution of Chinese language, the quantitative modeling and reconstruction of polysyllablization remain open questions. Based on the *Comprehensive Dictionary of Chinese Words*, a mapping from the words to their time of occurrence is built. With the inverse mapping on random samples, the newly produced words with different numbers of syllables in different time periods are obtained. Finally the total quadratic variation minimization model is adopted to estimate the trend of polysyllablization. As a novel exploration in the computational linguistics, the results agree with the stage division of historical Chinese and answer some difficult questions related to polysyllablization in a quantitative manner.

1. Introduction

Language is always changing [1]. The change of language is variation over time in a language's lexical, phonetic, syntactic, and other features. Old English (from the mid-5th century to the mid-11th century), for instance, would be greatly different from Modern English (from the late 15th century to the present) [2]. Language changes in many and varied ways. It is interesting to note that lexical change is one of the most obvious and important types of language change. One may observe that new words are formed and old words are tagged as "obsolete." Slang terms, in particular, come and go every few years. Consequently, as far as a long historical period is concerned, the variation of lexis is a complicated issue, which is not only fundamental to historical lexicology but also related to many research fields ranging from historical linguistics [3], onomasiology, etymology, to sociolinguistics [4].

Polysyllablization is one of the fundamental trends in the development of Chinese lexis. It is a prevailing view nowadays in the academic circle that there are three stages in the evolution of Chinese [5–8], that is, Old Chinese, Middle Chinese, and Modern Chinese. Old Chinese was the commonly used language during the early and middle Zhou Dynasty [9] (around 1046–256 BC). Middle Chinese was the historical Chinese dialect which was phonologically recorded in *Qieyun* [10, 11], a rime dictionary first published in 601. Modern Chinese is mainly referred to as the form of Chinese varieties to the present. However, the boundaries of these stages are ambiguous. The periods between them are transitional phases. The vocabulary system of Old Chinese mainly consists of monosyllabic (i.e., single syllable) words, whereas that of Modern Chinese is mainly made up of polysyllabic (i.e., two or more syllables) words. The aforementioned evolution of Chinese lexis is called polysyllablization. Since each Chinese character represents a monosyllabic Chinese word or a morpheme, polysyllablization in the historical evolution of Chinese not only introduces the form expansion of words but also represents the combination of morphemes and meanings and depicts the coordination of grammatical relations, which reflect the inherent laws of Chinese phonetics, semantics, and syntactics. In addition, the importance of syllables in Chinese lies in their link to word recognition [12, 13]. Therefore, polysyllablization is one of the most principal research topics of Chinese language, which has attracted lots of attention from the academia.

Most of the traditional studies on the polysyllablization of Chinese lexis are qualitative, of which the conclusions are mainly based on personal experiences. In fact, the difference in the number of syllables between Old Chinese words and Modern Chinese words is obvious and easy to be noticed. However, the following questions are not easy to answer based on the qualitative conclusions.

(i) What was the quantitative degree of polysyllablization during a certain period, for example, Qin or Han Dynasty? When did the polysyllabic words begin to occupy a major position in the vocabulary system?

(ii) What was the speed of polysyllablization at a certain time point? When was the lexis polysyllablized most rapidly?

(iii) How were the evolving trends for different kinds of polysyllabic words (i.e., disyllabic words, three-syllable words and so forth)?

As a result, the related studies in recent years have put a particular emphasis on the adoption of quantitative methods. They usually focus on a specific range of language materials, for example, a specific chapter, a specific book, or all the books of a specific author. For these limited language materials, the numbers of monosyllabic words, polysyllabic words, and total words have been counted. The ratio of the number of polysyllabic words to that of total words, namely, polysyllabic word ratio, is then adopted to evaluate the degree of polysyllablization. In a sense, polysyllablization is studied in a quantitative manner. Although the statistics on the polysyllabic words in a specific range of language materials are closely related to polysyllablization, it is not the polysyllablization itself. The polysyllabic word ratio of a specific range of language materials seriously depends on the selected topic, the author's habit, and the length of the corresponding text materials, so it is only a local weight of polysyllabic words and cannot represent the global degrees of polysyllablization. However, it is indeed very difficult to count all words of various numbers of syllables through different time periods for the purpose of depicting the trend of polysyllablization.

The lexis, that is, the vocabulary system, of a language is related to many complex and dynamic features with lots of uncertainties. It is often very difficult to construct a mathematical model. To the best of our knowledge, the quantitative evaluation and modeling of the polysyllablization of Chinese are still open questions.

Hanyu Da Cidian [14] (literally *Comprehensive Dictionary of Chinese Words*, abbreviated as *CDCW* in the following text) is the most inclusive Chinese dictionary available. It has a diachronic coverage of the Chinese language, tracing its usage over three thousand years from Chinese classic texts to modern slangs. Researchers use *CDCW* as a reference book to search and examine specific words and develop related lexicological studies. At the macrolevel, however, the enormous amounts of information in *CDCW* have not yet been effectively mined. To draw an analogy, if we compare *CDCW* to a building, the existing studies are focusing on the bricks; however, the structure of the building as a whole has been neglected. The data mining of *CDCW* provides us with a novel approach to tackle the polysyllablization of Chinese language.

This paper aims to answer the aforementioned open questions related to polysyllablization. In particular, our contribution lies mainly in the following aspects.

(i) We propose a novel approach based on *CDCW* to study polysyllablization of the Chinese lexis in a quantitative and impersonal manner. The words in *CDCW*, as a whole, are regarded as the maximum vocabulary so far. We can obtain the time of occurrence for each word in *CDCW*. Once a mapping from the words to their time of occurrence is built, the inverse mapping leads us to the newly produced words with different numbers of syllables in different time periods.

(ii) We introduce several techniques in the practical workflow to handle various uncertainties, including the repetitions and exceptions in the entries of *CDCW* and inaccuracy of mapping from words to their time of occurrence. The statistical sampling method is adopted to avoid the huge workload to process hundreds of thousands of words. The total quadratic variation minimization model is adopted to estimate the trend of polysyllablization.

(iii) We obtain the statistical data and finally present the trend of polysyllablization of the Chinese lexis. We also discuss the results through comparison with the existing conclusions. As a novel exploration in the computational linguistics, especially for the Chinese language with such a long history and such a complex evolution, our approach is valuable to similar issues.

The rest of this paper is organized as follows: Section 2 briefly reviews related works. Section 3 introduces the formulation and the mathematical model. Section 4 presents the results and discusses the trends of polysyllablization. Finally, Section 5 concludes this paper.

2. Related Works

Since polysyllablization is among the most important rules in the development of Chinese vocabulary, in recent years, studies on Chinese polysyllabic words and polysyllablization have achieved rich results. For example, the syllabic structure of Mandarin Chinese is discussed in [15] based on the X-bar approach. From the viewpoint of the Buddhist and Taoist scriptures of Eastern Han Dynasty, the polysyllablization of Chinese vocabulary based on the new lexical items is discussed in [16]. Since most of polysyllabic words are disyllabic words, the development of disyllabic words in Chinese is discussed in [17]. Also, many quantitative surveys and analyses have been conducted on polysyllabic words in different monographs across different time periods. According to the statistics in [18], published papers on the vocabulary in Middle Chinese and Early Modern Chinese monographs have amounted to over 6,000. Generally, without uniform standards, scholars differ from each other on their subjective definitions of polysyllabic words; as a result, they have obtained different statistical results about the number of the polysyllabic words in the same monograph. Take *Shi Shuo Xin Yu* (literally *A New Account of the Tales of the World*) for example, the number of polysyllabic words in it was said to be around 1,500 to 2,100 in different conclusions. Besides, due to the limitations of the subject, content, length of the book, and the authors mastery over the language, the vocabulary in one monograph alone cannot fully represent the complete picture of the its contemporary vocabulary system; as a matter of fact, sometimes there could be a great deviation. In other words, monographs are theme-based and scope-limited, so they can only reflect the vocabulary system from a restricted view rather than represent the whole. So far, many questions still remain unanswered, such as how polysyllablization evolved in the history of the Chinese vocabulary system.

CDCW is among the most important dictionaries in Chinese lexicological studies. Since the publication of its first volume in 1986, it has been put into wide academic application and achieved plentiful research results. *CDCW* has 50 million characters, containing 22.7 thousand Chinese characters and 375 thousand polysyllabic words. Wenkan Xu, one of the editorial board members of *CDCW*, said "Before the publishing of *CDCW*, there was no such a dictionary which contains both modern and ancient words, as well as words that were developed intermediately, and proves itself to be a confluence of the entire Chinese vocabulary, for the purpose of search and reference" (translated from Chinese according to [19]). Thus it is feasible for researchers either to observe the change of a single word from a diachronic perspective or to observe the semantic structure of a group of words from a synchronic perspective. Additionally, such a large-scale historical Chinese dictionary, which collects, organizes, and explains hundreds of thousands of ancient and modern words, is a very valuable corpus itself. Corpus linguistics is one of the most important fields nowadays [20]. *CDCW* is undoubtedly a huge corpus, and the reasonable utilization of *CDCW* will probably lead us to a novel approach, excluding the subjective prejudice of the researchers, to tackle the polysyllablization of Chinese.

3. Formulation and Mathematical Model

The vocabulary system has two characteristics, namely, integrity and dynamics. Integrity means that the vocabulary system is a macrolevel concept and refers to the sum of all the words in the language. Dynamics means that the vocabulary changes over time. Equivalently, new words appear and old words fade. Thus, in order to illustrate the polysyllablization of Chinese lexis, two key issues must be solved in accordance with these two characteristics of the vocabulary system. The first issue is to obtain the complete set of the words from Old Chinese to Modern Chinese. The second one is to determine the first-appearing and last-appearing time points of each word.

Unlike the English text in which sentences are sequences of words delimited by spaces, the word boundary in Chinese is fuzzy. A Chinese word may consist of one, two, or more Chinese characters (also referred to as *Hanzi*). There is no immediate way of deciding which characters in the text should be grouped into words. The studies on word segmentation have attracted lots of attention from the academia [21–23]. Once the words are obtained, the numbers of syllables are easy to be counted.

To assert the first-appearing and last-appearing time points of each word is complicated. A group of Chinese characters is possibly treated as a word at some time, but not considered as a word at another time. There are also various criteria to decide whether a group of characters is a word, most of which are personally prejudiced with lots of uncertainties.

We introduce a novel method to solve the aforementioned issues and analyze the polysyllablization of Chinese lexis in an impersonal and quantitative manner based on the data mining of *CDCW*. Briefly, we build a mapping from the words to their time of occurrence. The inverse mapping is used to count the newly formed words over different time periods. However, there are still lots of uncertainties to deal with. Thus we introduce a total quadratic variation minimization model and some related techniques to estimate the polysyllablization of Chinese lexis. The aforementioned issues finally lead to a constrained quadratic programming problem. The details are as follows.

3.1. Basic Idea. *CDCW* is the archive of Chinese vocabulary over time, containing 22.7 thousand Chinese characters (most of them are monosyllabic words) and 375 thousand polysyllabic words. Based on the entries of *CDCW*, the whole set of Chinese words over time can be obtained.

The structure of hundreds of thousands of entries in *CDCW* contains rich information and reflects many lexicological rules when examined at the macrolevel. Each entry has multiple properties, such as its pronunciation and explanation, which are directly listed in *CDCW*. In fact, more properties can be obtained indirectly with additional procedures. Among the indirect properties, the time of occurrence, which indicates when the word emerged, is of great significance to the studies of historical lexicology.

In *CDCW*, the definition of an entry usually contains several terms of meanings, most of which can be traced back to their earliest reference through the documentary evidences provided by *CDCW*. As long as the approximate time when the documentaries were published or the years when their authors lived are possible to be acquired, we will be able to date all the documentaries available for each term of meanings for the word entry, the earliest one of them reveals the time of occurrence of the word, that is, the time when the word emerged. Note that *CDCW* is a result of collective wisdom of more than 1000 scholars. Thus it offers us an authoritative and impersonal manner to deduce the time of occurrence for each word.

Here is an example to illustrate the procedure from the entry to its time of occurrence. For the word entry *Senlin* (Chinese word, literally, forest; here Chinese characters *Sen* and *Lin* both mean trees or woods), there is only one term of meaning. The earliest documentary evidence is a poem in Tang Dynasty. Thus, it is deduced that the word *Senlin* was composited during Tang Dynasty. We also know the author, Xiji Cai, served as a junior officer in Luoyang City in 748. However, we know neither the exact year when he wrote the poem nor the birth year of him.

The inaccuracy in defining the publishing time of the ancient documentaries is not occasional. In fact, limited by the records of ancient history literatures, the exact years of the publication of ancient documentaries or even the exact living years of the related authors are difficult to resolve. For most of ancient documentaries, we only know the dynasties of their publication.

Figure 1 is a diagram showing how to investigate the time of occurrence entry by entry in *CDCW*. Suppose that the earliest documentary evidence for Entry 1 is found in Wei Dynasty, then its time of occurrence can be determined to be Wei Dynasty and it will be put under the corresponding chronological category. The same procedure applies to the rest of the entries. In theory, following this method, the time of occurrence for all entries can be determined exhaustively.

The above process is to determine the time of occurrence per word entry by exploring its earliest documentary evidence. However, in order to observe the dynamic evolution of the vocabulary system at the macrolevel, it is necessary to conduct another survey from the opposite direction, that is, to investigate words which occurred in the same time period. By gathering all words which share the same time span of occurrence in *CDCW*, we can obtain all the new words that emerged in the corresponding time period.

The evolution of the vocabulary system is a metabolic process. On one hand, new words have been emerging constantly and entering the vocabulary system; on the other hand, some old words have been withering away and falling out of the vocabulary system. The emergence of new words usually reflects the reform of the society, the occurrence of new things, the transformation of concepts, the evolution of the language itself, and so forth. The emergence of new words is an active and positive factor in the development of the vocabulary system and constitutes the subject of most lexicological studies. In fact, even when we talk about the fading of the old words, they do not completely exit but still exist as a historical preservation in the vocabulary system. Considering that the emergence of new words takes a much more important position than the extinction of old words, we can obtain the general trend of the diachronic evolution of Chinese vocabulary system at the macrolevel by investigating the new words categorized in different time periods according to their time of occurrence in *CDCW*.

Admittedly, omissions of entries and presence of errors can be found in *CDCW*, and it is also true that *CDCW* cannot include all the new words in a given time period. However, the definitions and explanations of words in *CDCW* are the concerted efforts of a great number of researchers and scholars, and the defects and mistakes in it only account for a trivial proportion in the total number of its entries. As the "archives of ancient and modern Chinese vocabulary," *CDCW* remains one of the most authoritative dictionaries. Therefore, it is not only feasible, but also conducive to excluding the subjective prejudice of the researchers and to use the time of occurrence per entry in *CDCW* for reference when studying the diachronic evolution of the Chinese vocabulary system.

Theoretically speaking, it is possible to exhaustively obtain the time of occurrence of all word entries in *CDCW*. However, considering the fact that *CDCW* contains as many as over 300 thousand entries, it would be too much work to analyze all the entries. Therefore, in the present study, under the guidance of statistical sampling theory, we only draw a moderate number of sample entries and then deduce the overall quantitative characteristics of the population.

3.2. Formulation under Ideal Condition. Let W denote the word set for the Chinese lexis, from Old Chinese to Modern Chinese. An element $w \in W$ is a Chinese word. The subsets of monosyllabic words, disyllabic words, and three-or-more-syllable (i.e., ≥3 syllables) words are written as $W^{(1)}$, $W^{(2)}$, and $W^{(3+)}$, respectively. The subset of polysyllabic words is written as $W^{(2+)}$ and satisfies $W^{(2+)} = W^{(2)} \cup W^{(3+)}$. Notice that disyllabic words are processed independently because disyllabic words are extremely important in Chinese. Let the cardinality of a set be denoted as $|\cdot|$. Thus the total number of the words is represented as $N = |W|$. Likewise, the number of words in $W^{(1)}$ is denoted as $N^{(1)} = |W^{(1)}|$ and so forth.

Let T represent the time (using year as the unit of time) for the evolution of Chinese language, which is in fact a range of integer number from t_b to t_e. The time length in years of T is given by $\|T\| = t_e - t_b + 1$.

As illustrated in Figure 1, there exists a function f which relates each word $w \in W$ to its time of occurrence $t = f(w) \in T$. With the denotation, $f^{-1}(t)$ represents all the words which first occurred in the year t. The symbol $n_t = |f^{-1}(t)|$ represents the total number of the words which first occurred in the year t. Considering that n_t is the number of the newly formed words in the year t, we define it as the speed of new word production.

The sequence $\{n_t\}$ constitutes a time series. Its partial sum sequence $\{c_t\}$ is the cumulative number of all the words which first occurred in or before the year t, satisfying

$$c_t = c_{t_b^-} + \sum_{\tau=t_b}^{t} n_\tau. \qquad (1)$$

Here, $c_{t_b^-}$ means the initial cumulative number of all the words before time t_b.

In addition, let $f|_U : U \to T$ denote the restriction of f to the subset U of its domain W, which is defined by the same rule as f but with a smaller domain set U. Thus, $n_t^{(1)} = |f|_{W^{(1)}}^{-1}(t)|$ represents the total number of the monosyllabic words which first occurred in the year t. Likewise, we have $n_t^{(2)}$, $n_t^{(3+)}$ and also their partial sum $c_t^{(2)}$, $c_t^{(3+)}$, and so forth.

We introduce two indexes to evaluate the trend of polysyllablization. The first is the speed of new word production for polysyllabic words, which is, namely, $n_t^{(2+)}$. It is in fact the number of the newly formed polysyllabic words per year and indicates how fast the new polysyllabic words are generated into the vocabulary system. The second index is named the polysyllablization degree index (PDI). It is the ratio of the cumulative number of the polysyllabic words to that of all the

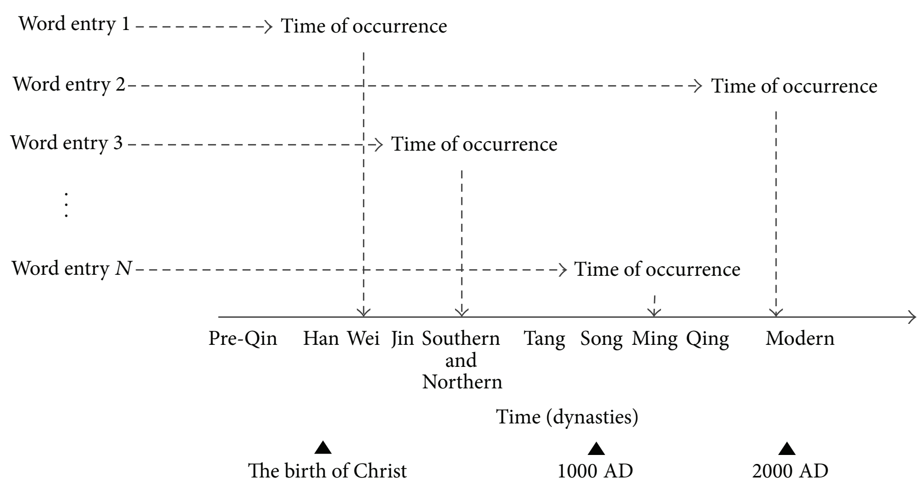


FIGURE 1: Determining the time of occurrence per word entry by the earliest textual evidence with *CDCW*.

words at the time t, and indicates the weight of the polysyllabic words in the then vocabulary system, which is defined as

$$\text{PDI}(t) \triangleq \frac{c_t^{(2+)}}{c_t}. \tag{2}$$

3.3. Modeling with Uncertainties. However, there are still lots of uncertainties in the aforementioned formulation.

Firstly, there are a small amount of repetitions and exceptions in the entries in *CDCW*. The repetitions mean that two entries have different forms but the same meanings, which is similar to the case "color" and "colour" in English. Basically, they are just different forms of one word and should be counted as one word only. The exceptions mean that some entries are not valid Chinese words, such as the Japanese characters included in *CDCW*. Thus, the set of the whole entries of *CDCW*, E, is a superset of W; that is, $W \subset E$. The total number of entries in E is greater than that in W. The invalid entries, including repetitions and exceptions, should be preprocessed.

Secondly, there are above 300 thousand of entries in *CDCW*. They are too many to be handled one by one. The feasible way is to handle part of them with the statistical sampling method, which subsequently introduces probabilistic uncertainty. Assuming that we take a certain number of samples which are uniformly randomly chosen from the entries of *CDCW*, the set of the sampled entries is denoted as E_s, in which the set of valid entries (i.e., words) is denoted as W_s.

Based on the samples, the total number of word in *CDCW* can be estimated by

$$\widehat{N} = \widehat{|W|} = \frac{|W_s|}{|E_s|} \cdot |E| = \frac{|E|}{|E_s|} \cdot |W_s| = K \cdot |W_s|. \tag{3}$$

Here K is a constant factor and equals $|E|/|E_s|$.

Furthermore, the sequence $\{n_t\}$ can be estimated by the restriction of f to the subset W_s, which is similarly given by

$$\widehat{n}_t = K \cdot \left| f|_{W_s}^{-1}(t) \right|. \tag{4}$$

The cases to estimate $\widehat{n}_t^{(1)}$, $\widehat{c}_t$, and so on are similar and not listed here.

Thirdly, the precise value of the function f is usually difficult to decide. For most of the words, we can only know the dynasty of its occurrence rather than the time (accurate to year) of its occurrence. Thus the inaccuracy of time of occurrence will introduce lots of uncertainties.

Now consider that the set T can be divided into several segments, mathematically speaking, T can be expressed as the union of a number of disjoint sets D_i $(1 \le i \le n_d)$ as follows:

$$T = \bigcup_{1 \le i \le n_d} D_i, \tag{5}$$

where D_i is a set of consecutive integers beginning with $t_b^{(i)}$ and ending with $t_e^{(i)}$, that is, an integer interval with notation $[t_b^{(i)} \cdots t_e^{(i)}]$. The symbol n_d represents the number of the segments. Let D denote the set $\{D_1, D_2, \ldots, D_{n_d}\}$.

In fact, D_i represents a dynasty or a similar time period in Chinese history. Such partitions are caused by the uncertainty of the mapping from the words to their times of occurrence. The exact time of occurrence of many words is difficult or even impossible to resolve due to the obscureness in the ancient documentaries. Since dates are usually written in dynasties or reign titles in the ancient Chinese literatures, the exact dynasties (or time segments) can be determined for the first occurrence of many words. Thus the dynasties are the ordinary and natural implementation of the time segments.

Without loss of generality, we assume that the segments D_i are sorted by the lower endpoint, which satisfies

$$\begin{aligned} t_b^1 &= t_b, \\ t_e^i + 1 &= t_b^{i+1}, \quad \text{where } 1 \le i \le n_d - 1, \\ t_e^{n_d} &= t_e \end{aligned} \tag{6}$$

.

The rough version of the function f, which only maps from the word set W to the set of time segments (usually, dynasties) D, is written by

$$g : W \longrightarrow D \tag{7}$$

and satisfies

$$g(w) = d \Longleftrightarrow f(w) \in d. \tag{8}$$

Since the precise mapping from the words to their time of occurrence is usually unknown in the actual operation, we must estimate the parameters related to polysyllablization with the rough function g. It can be noted that $g^{-1}(D_i)$ represents all the words which firstly occurred in the time segment D_i. In fact, there exits the following relation:

$$g^{-1}(D_i) = \bigcup_{\tau \in D_i} f^{-1}(\tau), \quad \text{where } 1 \le i \le n_d. \tag{9}$$

Now that $f^{-1}(\tau)$ and $f^{-1}(\nu)$ are disjoint sets if $\tau \ne \nu$, we have

$$d_i = \left|g^{-1}(D_i)\right| = \sum_{\tau \in D_i} \left|f^{-1}(\tau)\right| = \sum_{\tau \in D_i} n_\tau, \quad \text{where } 1 \le i \le n_d. \tag{10}$$

Here, d_i represents the number of all the words which first occurred in the time segment D_i.

Based on the statistical sampling, the sequence $\{d_i\}$ can be estimated by the restriction of g to the subset W_s, which is similarly given by

$$\widehat{d}_i = K \cdot \left| g|_{W_s}^{-1}(t) \right|. \tag{11}$$

Let $d_i^{(1)}$ be the number of all the monosyllabic words which first occurred in the time segment D_i and so on. The cases to estimate $d_i^{(1)}$, $d_t^{(2)}$, and so forth are similar and not listed here.

Language changes over time. As a complex social phenomenon, language may change smoothly or unstably. We know little about the actual trends of the evolution of language. Without any prior knowledge or assumption, we will build a total variation minimization model [24, 25] for the evolution of the language. Other methods, such as maximum entropy [26, 27] method for the time series, should also be feasible and constructive. However, only the total variation minimization model is discussed in this paper, since it is simply based on the assumption that the language of today is not far from that of yesterday. Thus it is probably one of the safest estimation methods to start our exploration in such an open research field.

The total quadratic variation for a sequence $x = \{x_i\}$ $(1 \le i \le n)$ is defined as

$$V(x) \triangleq \sum_{i=1}^{n-1} \left(x_{i+1} - x_i\right)^2. \tag{12}$$

Our technique to estimate the sequences $n_t^{(1)}$, $n_t^{(2)}$, and $n_t^{(3+)}$ are similar. Generally, the case for the sequence $n_t^{(*)}$ is illustrated as follows, in which the symbol "$*$" can be replaced by any one of the symbols including "1," "2," or "3+." Basically, the issue to estimate the sequence $n_t^{(*)}$ is equivalent to a constrained optimization problem as follows:

$$\begin{aligned} &\text{minimize} \quad V\left(n^{(*)}\right) = \sum_{\tau = t_b}^{t_e} \left(n_{i+1}^{(*)} - n_i^{(*)}\right)^2 \\ &\text{subject to} \quad n_t^{(*)} \ge 0, \quad \text{where } t_b \le t \le t_e, \\ &\qquad \sum_{\tau = t_b^{(i)}}^{t_e^{(i)}} n_\tau^{(*)} = d_i^*, \quad \text{where } 1 \le i \le n_d. \end{aligned} \tag{13}$$

Here, the latter constraints can be similarly deduced as (10). Such a constrained optimization problem is a quadratic programming problem, which belongs to a relatively mature area and there are kinds of feasible techniques [28–30] to handle it. The problem can be solved in polynomial time with the ellipsoid method [31].

3.4. Work Flow. There are a series of steps to evaluate polysyllablization of the Chinese lexis based on *CDCW* in practice.

(1) Decide the endpoints of the time T and split the whole time into segments D_i.
(2) Obtain the set E of the whole entries of *CDCW* and classify the set E into $E^{(1)}$, $E^{(2)}$, and $E^{(3+)}$ by the number of syllables of each entry.
(3) Draw appropriate samples of entries, that is, $E_s^{(1)}$, $E_s^{(2)}$, and $E_s^{(3+)}$ from the sets $E^{(1)}$, $E^{(2)}$, and $E^{(3+)}$, respectively.
(4) Check each element e in the sets $E_w^{(1)}$, $E_s^{(2)}$, and $E_s^{(3+)}$ to obtain the sets of word samples, $W_s^{(1)}$, $W_s^{(2)}$, and $W_s^{(3+)}$.
(5) Estimate the total number of monosyllabic words $\widehat{N}^{(1)}$. Estimate $\widehat{N}^{(2)}$, $\widehat{N}^{(3+)}$ in the same way. Calculate the total number of words $\widehat{N} = \widehat{N}^{(1)} + \widehat{N}^{(2)} + \widehat{N}^{(3+)}$.
(6) Map each word in $W_s^{(1)}$, $W_s^{(2)}$, and $W_s^{(3+)}$ to its time of occurrence.
(7) Count the number of words in the same time segment and estimate $\widehat{d}_i^{(1)}$, $\widehat{d}_i^{(2)}$, and $\widehat{d}_i^{(3+)}$ according to (11).
(8) Solve the quadratic programming problem to obtain the estimation $\widehat{n}_t^{(1)}$, $\widehat{n}_t^{(2)}$, and $\widehat{n}_t^{(3+)}$.
(9) Calculate $\widehat{n}_t^{(2+)} = \widehat{n}_t^{(2)} + \widehat{n}_t^{(3+)}$ and $\widehat{n}_t = \widehat{n}_t^{(1)} + \widehat{n}_t^{(2+)}$. Also, calculate their partial sum sequences.
(10) Calculate the indexes to evaluate the trend of polysyllablization.

4. Analysis of the Polysyllablization

We use the CD-ROM version of *CDCW* published in 1998. This edition contains 27,989 character entries, 279,720 disyllabic word entries, and 63,587 three-or-more-syllable word

Table 1: Statistical data of words of various numbers of syllables.

i	Dynasties	Endpoints	Single syllable		Double syllables		Three or more syllables	
	d_i	$t_b^{(i)}$–$t_e^{(i)}$	$\widehat{g}_i^{(1)}$	$\widehat{d}_i^{(1)}$	$\widehat{g}_i^{(2)}$	$\widehat{d}_i^{(2)}$	$\widehat{g}_i^{(3+)}$	$\widehat{d}_i^{(3+)}$
1	pre-Qin	1045 BC–222 BC	274	7669	246	34406	55	3497
2	Qin, Western Han	221 BC–24 AD	44	1231	135	18881	33	2098
3	Eastern Han	25–219	56	1567	128	17902	22	1399
4	Three kingdoms, Jin	220–419	12	336	115	16084	23	1462
5	Southern and Northern	420–580	44	1231	238	33287	48	3052
6	Sui, Tang	581–907	23	644	275	38461	97	6168
7	5 dynasties, Song	908–1279	138	3862	243	33986	110	6995
8	Yuan, Ming, Qing	1280–1949	37	1036	391	54685	306	19458
		Total		17576		247692		44129

entries, from which we draw 1000 character entries, 2,000 disyllabic word entries, and 1,000 three-or-more-syllable word entries. Here we perform uniformly random sampling on all the entries without replacement. We use python, a computer script language, to perform the data sampling and processing.

The number of effective samples of monosyllabic words is 669 (66.9% of the drawn sample entries). According to the statistical sampling theory, the total number of effective monosyllabic words in *CDCW* is estimated to be about $27,898 \cdot 66.9\% \simeq 18,724$. The number of effective samples of the disyllabic words is 1,904 (95.2% of the drawn sample entries) and that of the words with three or more syllables is 876 (87.6% of the drawn sample entries). Likewise, the total number of disyllabic words in *CDCW* is estimated to be about $279,720 \cdot 95.2\% \simeq 266,293$ and that of three-or-more-syllable words is about $63,587 \cdot 87.6\% \simeq 55,702$. The total number of polysyllabic words in the Chinese lexis is about $266,293 + 55,702 = 321,995$. Thus, the number of polysyllabic words is greatly more than that of monosyllabic words up to now. Disyllabic words constitute the major part of the whole vocabulary.

We set the endpoints of the time axis to 1045 BC and 1949 AD. The former is approximately the beginning of Zhou Dynasty. The latter is the founding of the People's Republic of China. The time axis is split into 8 segments. We map all these effective word samples to their time of occurrence and count the number of samples belonging to each time segment.

The statistical data are listed in Table 1. The words which were newly formed after 1949 are ignored to avoid boundary problems. Therefore the sum in the table is a little less than the total number of words.

The data in Table 1 show the total number of the newly emergent words in each time period. For example, there were about 33,287 disyllabic words produced in Southern and Northern Dynasties and 38,461 in Tang Dynasty, the latter exceeding the former. However, the span of the former time segment was only 160 years and that of the latter was as long as 327 years. If we distribute the newly emergent words evenly among the years, the annual average for the Southern and Northern Dynasties is $33287/161 \simeq 206.8$ (word per year) and that for Tang Dynasty is $38461/327 = 117.6$ (word per year), only about half of the number in Southern and Northern Dynasties.

Notice that there were about 7,669 monosyllabic words produced in the pre-Qin period (1045 BC–222 BC). Part of them are generated during or before Shang Dynasty and known as Oracle Bone Script [32]. Several thousand bones and plastrons have been reconstructed and many thousands of texts have been studied. The texts contain over 30,000 distinct characters, which are thought to be variant forms of around 4,000 individual characters. Thus, the number of characters before Zhou Dynasty should be regarded as the initial cumulative number of the monosyllabic words. Now $c_{t_b}^{(1)}$ is set to be 4000, that is, the number of monosyllabic words at the beginning of pre-Qin period. We also subtract $c_{t_b}^{(1)}$ from the estimated number of the monosyllabic words in the pre-Qin period thereafter so as to make a revised estimation. The estimation on polysyllabic word is not revised because it is still argumentative whether there existed polysyllabic words in the Oracle Bone Script.

Figure 2 shows the final estimation on the speed of new word production. It illustrates that the speed is changeable. The speed of the new polysyllabic word production $n_t^{(2+)}$ is above that of monosyllabic words $n_t^{(1)}$ from the beginning of the time axis. It can be seen that the speed of new polysyllabic word production increased and reached two peaks at about 75 AD and 550 AD, that is, the Eastern Han Dynasty and Southern and Northern Dynasties. In the following years, new polysyllabic words still emerged, yet at a slower pace through Song, Yuan, Ming and Qing Dynasties. In fact, Southern and Northern Dynasties are treated as the center of Middle Chinese. Figure 2 shows that the Middle Ages are the critical transition period of polysyllablization. During Eastern Han and the following dynasties, the cultures from the western neighbors of China had a tremendous influence upon not only the Chinese traditional culture but also the Chinese language [33]. Linguistic evidences are available to reveal the early cultural exchange between China and India [34]. For example, the language contact [35] caused by the Buddhist texts translated from foreign languages (mainly Sanskrit) into Chinese was an important exterior influential factor to the evolution of the Chinese language. From this

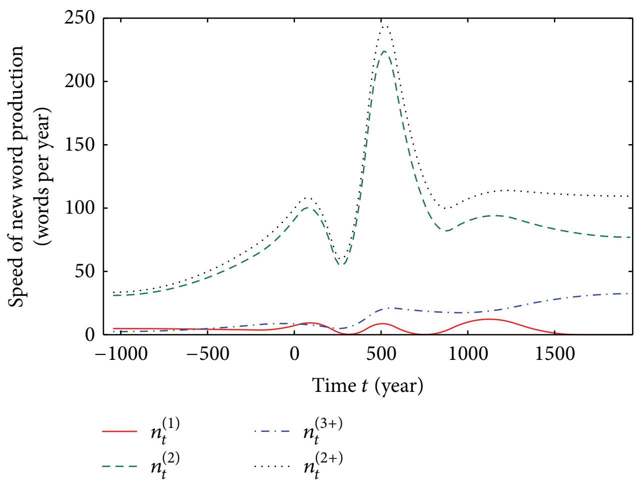


FIGURE 2: The speed of the word production.

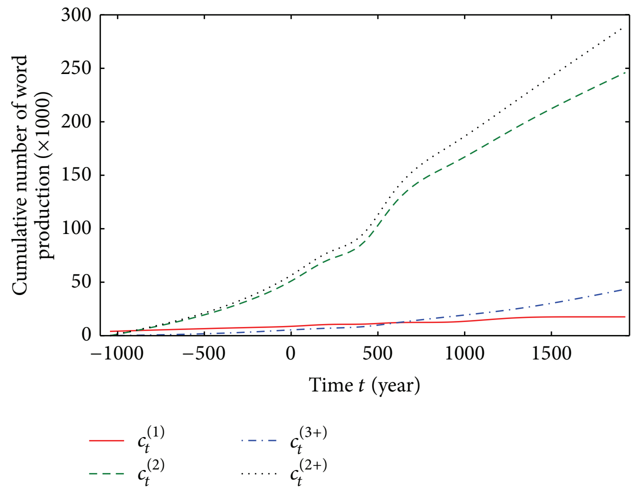


FIGURE 3: The cumulative number of the word production.

perspective, it can be concluded that the rising and falling in the above curves do reflect the evolution inside the Chinese language, agree with its historical stage division, and support the findings in other studies on the historical linguistics.

Figure 3 shows the final estimation on the cumulative number of new word production. It illustrates the increasing trend of the volume of Chinese vocabulary system. It is interesting to note that the amount of polysyllabic words $c_t^{(2+)}$ exceeds that of monosyllabic words $c_t^{(1)}$ before the ending of the first time segment. Thus polysyllabic words already were the major component of the lexis even in Old Chinese. However, back then, their frequency of use was very low, which has misled researchers to reach wrong judgments about the actual degree of the polysyllablization.

The trend that polysyllabic words became the major component of the Chinese lexis is also shown in Figure 4. It indicates that the weight of polysyllabic words in the Chinese lexis increased all along. At the ending of the time axis (1949 AD), about 94% of the Chinese words were polysyllabic.

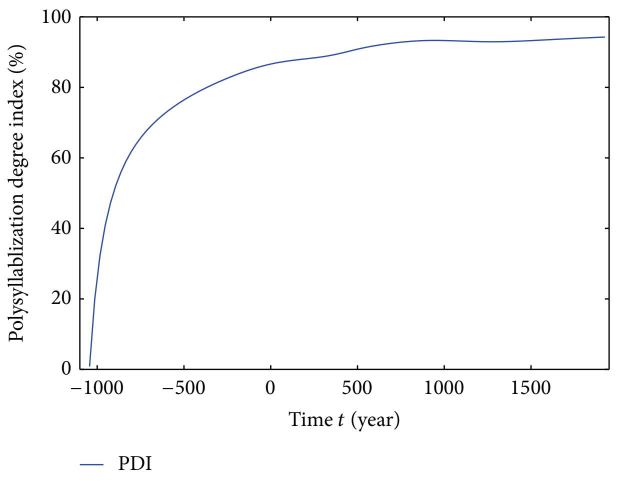


FIGURE 4: The quantitative trend of the polysyllablization degree index (PDI).

5. Conclusion

Polysyllablization is among the most important rules in the development of the Chinese lexis. Furthermore, due to the peculiarity of Chinese language, polysyllablization is not only a key issue in lexis, but also closely related to phonetics, semantics, and syntactics. However, the existing studies on polysyllablization are either in a qualitative manner or limited to a quantitative survey of specific language materials. The quantitative trend of polysyllablization is yet to be explored.

The lexis itself is an integral, dynamic, and complex system. With lots of uncertainties, its historical evolution is difficult to trace. As a novel exploration in the computational linguistics, we try to reconstruct the quantitative trend of polysyllablization of the Chinese lexis. A mapping from the words to their time of occurrence is built based on *Comprehensive Dictionary of Chinese Words*, a large-scale historical Chinese dictionary, which collects, organizes, and explains hundreds of thousands of ancient and modern words. Related formulation, mathematical model, and corresponding algorithms are introduced. We finally deduce the reconstruction to a constrained optimization problem based on the statistical sampling data. Such solution is really a data-mining procedure on the dictionary. It is not only feasible, but also conducive to excluding the subjective prejudice of the researchers. The results agree with the stage division of historical Chinese and answer some difficult questions related to polysyllablization in a quantitative manner.

Our research is only an initiatory exploration in this open research field. Our future research will focus on the revised mathematical models and algorithms, including statistical inference, maximum entropy estimation, and fuzzy mathematical theory based on more samples of word entries.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

The authors would like to thank the anonymous reviewers for their valuable comments and suggestions to improve the quality of the paper. The authors also would like to thank Ms. Lu Yang and Ms. Di Lu for their assistance in the preparation of this paper. This work was supported by Beijing Higher Education Young Elite Teacher Project (Grant no. YETP0869) and Beijing Fund for the Humanities and Social Sciences (Grant no. 11WYC026).

References

[1] R. Lass, *Historical Linguistics and Language Change*, vol. 81, Cambridge University Press, 1997.

[2] R. M. Hogg and D. Denison, *A History of the English Language*, Cambridge University Press, 2006.

[3] B. D. Joseph and R. D. Janda, *The Handbook of Historical Linguistics*, Wiley Online Library, John Wiley & Sons, 2003.

[4] J. K. Chambers, *Sociolinguistic Theory: Linguistic Variation and Its Social Significance*, Blackwell, Cambridge, Mass, USA, 1995.

[5] E. G. Pulleyblank, "Lexicon of reconstructed pronunciation," in *Early Middle Chinese, Late Middle Chinese, and Early Mandarin*, UBC press, 1991.

[6] W. H. Baxter, *A Handbook of Old Chinese Phonology*, vol. 64, Walter de Gruyter, 1992.

[7] L. Sagart, *The Roots of Old Chinese*, vol. 184 of *Current Issues in Linguistic Theory*, John Benjamins, 1999.

[8] G. Edwin Pulleyblank, *Middle Chinese: A Study in Historical Phonology*, UBC Press, 2011.

[9] D. C. Twitchett, J. K. Fairbank, A. Feuerwerker, W. J. Peterson, K.-C. Liuv, and R. MacFarquhar, *The Cambridge History of China, Volume 1991*, Cambridge University Press, 1978.

[10] E. G. Pulleyblank, "Qieyun and yunjing: the essential foundation for chinese historical linguistics," *Journal of the American Oriental Society*, vol. 118, no. 2, pp. 200–216, 1998.

[11] L. Sagart, "The origin of Chinese tones," in *Proceedings of the Symposium/Cross-Linguistic Studies of Tonal Phenomena/Tonogenesis, Typology and Related Topics*, pp. 91–104, 1999.

[12] X. Zhou, W. Marslen-Wilson, M. Taft, and H. Shu, "Morphology, orthography, and phonology in reading Chinese compound words," *Language and Cognitive Processes*, vol. 14, no. 5-6, pp. 525–565, 1999.

[13] C. McBride-Chang, X. Tong, H. Shu, A. M.-Y. Wong, K. Leung, and T. Tardif, "Syllable, phoneme, and tone: psycholinguistic units in early Chinese and english word recognition," *Scientific Studies of Reading*, vol. 12, no. 2, pp. 171–194, 2008.

[14] Z. Luo, Ed., *Hanyu Da Cidian (Chinese Dictionary, Literally, Comprehensive Dictionary of Chinese Words)*, Hanyu Da Cidian Press, Shanghai, China, 1988, (Chinese).

[15] J. van de Weijer and J. Zhang, "An X-bar approach to the syllable structure of Mandarin," *Lingua*, vol. 118, no. 9, pp. 1416–1428, 2008.

[16] L. Yu and M. Gu, "Polysyllablization of chinese vocabulary based on the new lexical items in the buddhist and taoist scriptures of eastern han," *Journal of Sino-Western Communications*, vol. 5, no. 1, p. 225, 2013.

[17] S. Duanmu, "Stress and the development of disyllabic words in Chinese," *Diachronica*, vol. 16, no. 1, pp. 1–35, 1999.

[18] Z. Guo and Q. Yan, "A survey of vocabulary studies of special books in middle and modern Chinese," *Chinese Journal, Literally, Forward Position*, vol. 4, pp. 145–150, 2011 (Chinese).

[19] W. Xu, "Brief comments on the special features and worth of hanyu da cidian," *Cishu Yanjiu(Chinese Journal, literally, Studies on Dictionaries)*, vol. 1994, no. 3, pp. 36–45, 1994 (Chinese).

[20] D. Biber, S. Conrad, and R. Reppen, *Corpus Linguistics: Investigating Language Structure and Use*, Cambridge University Press, 1998.

[21] H. Li and B. Yuan, "Chinese word segmentation," in *Proceedings of the 12th Pacific Asia Conference on Language, Information and Computation*, pp. 212–217, Singapore, February 1998.

[22] N. Xue, "Chinese word segmentation as character tagging," *Computational Linguistics and Chinese Language Processing*, vol. 8, no. 1, pp. 29–48, 2003.

[23] J. K. Low, H. T. Ng, and W. Guo, "A maximum entropy approach to chinese word segmentation," in *Proceedings of the 4th SIGHAN Workshop on Chinese Language Processing*, vol. volume, pp. 161–164, Jeju Island, Republic of Korea, 2005.

[24] L. I. Rudin, S. Osher, and E. Fatemi, "Nonlinear total variation based noise removal algorithms," *Physica D: Nonlinear Phenomena*, vol. 60, no. 1–4, pp. 259–268, 1992.

[25] A. Chambolle, "An algorithm for total variation minimization and applications," *Journal of Mathematical Imaging and Vision*, vol. 20, no. 1-2, pp. 89–97, 2004.

[26] M. Palus, "Kolmogorov entropy from time series using information-theoretic functionals," *Neural Network World*, vol. 7, no. 3, pp. 269–292, 1997.

[27] C. Bandt and B. Pompe, "Permutation entropy: a natural complexity measure for time series," *Physical Review Letters*, vol. 88, no. 17, Article ID 174102, 2002.

[28] M. Frank and P. Wolfe, "An algorithm for quadratic programming," *Naval Research Logistics Quarterly*, vol. 3, pp. 95–110, 1956.

[29] P. Wolfe, "The simplex method for quadratic programming," *Econometrica*, vol. 27, pp. 382–398, 1959.

[30] P. T. Boggs and J. W. Tolle, "Sequential quadratic programming," *Acta Numerica*, vol. 4, no. 1, pp. 1–51, 1995.

[31] M. K. Kozlov, S. P. Tarasov, and L. G. Khachiyan, "The polynomial solvability of convex quadratic programming," *USSR Computational Mathematics and Mathematical Physics*, vol. 20, no. 5, pp. 223–228, 1980.

[32] D. N. Keightley, *Sources of Shang History: The Oracle Bone Inscriptions of Bronze Age China*, University of California Press, 1978.

[33] E. Zürcher, *The Buddhist Conquest of China: The Spread and Adaptation of Buddhism in Early Medieval China*, vol. 1, Brill Archive, 1959.

[34] Q. Zhu, *Some Linguistic Evidence for Early Cultural Exchange between China and India*, vol. 66 of *Sino-Platonic Papers*, Department of Asian and Middle Eastern Studies, University of Pennsylvania, 1995.

[35] S. G. Thomason and T. Kaufman, *Language Contact*, Edinburgh University Press, Edinburgh, UK, 2001.

Hindawi Publishing Corporation
Journal of Applied Mathematics
Volume 2014, Article ID 814159, 15 pages
http://dx.doi.org/10.1155/2014/814159

Research Article

Towards Light-Weight Probabilistic Model Checking

Savas Konur

Department of Computer Science, University of Sheffield, Sheffield S1 4DP, UK

Correspondence should be addressed to Savas Konur; s.konur@sheffield.ac.uk

Received 28 January 2014; Revised 19 May 2014; Accepted 3 June 2014; Published 10 July 2014

Academic Editor: Guiming Luo

Model checking has been extensively used to verify various systems. However, this usually has been done by experts who have a good understanding of model checking and who are familiar with the syntax of both modelling and property specification languages. Unfortunately, this is not an easy task for nonexperts to learn description languages for modelling and formal logics/languages for property specification. In particular, property specification is very daunting and error-prone for nonexperts. In this paper, we present a methodology to facilitate probabilistic model checking for nonexperts. The methodology helps nonexpert users model their systems and express their requirements without any knowledge of the modelling and property specification languages.

1. Introduction

Model checking [1] is a computational and algorithmic verification technique analysing if certain requirements hold in a system. These requirements are expressed as formal properties, such as *temporal logic* formulas. Model checking then exhaustively checks if these formal properties are satisfied by a structured model, for example, state transition system and finite state automaton, describing *all* system behaviours.

Model checking is an established research subject within computer science. There has been a vast amount of work carried out during the last two decades. Many tools have been devised and used in both academia and industry, for example, NuSMV [2], Spin [3], Uppaal [4], and Kronos [5]. Since model checking provides a comprehensive and an exhaustive computational analysis, it can reveal all possible system behaviours, which cannot be normally done by simulation or testing techniques. For this reason, it has been applied to various engineering problems, for example, hardware verification, software and programme verification, analysis of communication protocols, and safety-critical systems.

Standard model checking techniques are normally used to analyse *qualitative* temporal and dynamic properties. However, in order to have a deeper understanding of the system behaviour, some *quantitative* analysis is also required. Along with technological advances, systems are getting more complex. This requires the formal analysis to also include other aspects such as *uncertainy*, as the systems are getting more ubiquitous and probabilistic. In fact, while analysing many realistic systems, we need to consider uncertainty in system components and the reliability of communication [6] as well as uncertainty in planning and analysing performance. Unfortunately, standard model checking techniques are not sufficient to provide the level of analysis that we need in these cases.

Probabilistic model checking is a probabilistic variant of classical temporal model checking, which provides quantitative information regarding the likelihood that certain system behaviour is observed. This verification method has been successfully applied to formal analysis of a vast number of systems, for example, "bluetooth protocols, self-configuring protocols, self-organising systems, fault-tolerant algorithms, and scalable protocols" [7]. This is an active research subject within the model checking study. Various tools have been devised, such as Prism [8], Mrmc [9], Ymer [10], and Vesta [11]. Among these, Prism is the most widely used probabilistic model checking tool. Mrmc is also another popular tool with new improvements employed recently.

Although probabilistic model checking tools have been used to verify various systems, this usually has been done by experts who have a good understanding of model checking and who are familiar with the syntax of both modelling and property specification languages. Unfortunately, this is not an easy task for nonexperts to learn description languages for modelling and formal logics/languages for property specification. In particular, property specification is very daunting and error-prone for nonexperts. Research has shown that even experts make errors when they formally specify an informal requirement [12].

In this paper, we present a methodology to facilitate probabilistic model checking for nonexperts. As part of our methodology, we propose using some tools to achieve this. Namely,

(i) we propose a *property generator* tool which automatically generates formal properties using natural language statements;

(ii) we use a *front-end* tool which provides a graphical-user interface that allows constructing probabilistic state machines from which the model code is automatically generated;

(iii) we also devise a strategy which reduces the state space of probabilistic models to increase the performance of model checking and relieve the *state explosion* problem.

In this way, nonexperts can model their systems and express their requirements without any knowledge of the modelling and property specification languages and can analyse their systems using probabilistic model checking without having any expertise. We are not aware of any work which considers simplifying both modeling and property specification for model checking tool(s)—except very few studies introducing some simplifications for property specification, which will be reported in Section 5. We believe the idea and approach presented in this paper is novel, and this is an important step towards making model checking a more accessible computational technique for nonexpert users.

The paper is organised as follows: Section 2 summarises probabilistic model checking; Section 3 presents our methodology; Section 4 describes the model construction component; Section 5 describes the property generation component; Section 6 applies the methodology to an example system; Section 7 discusses how we will extend our methodology; and Section 8 concludes the paper.

2. Probabilistic Model Checking

In model checking, a finite system model (e.g., state transition system and finite state automaton), representing *all* system behaviours, is checked against a temporal logic formula. The entire state space represented by this model is then exhaustively analysed to verify if the formula is satisfied. For example, model checking allows us to check if the temporal logic formula,

$$\mathrm{AG}\neg\texttt{deadlock} \tag{1}$$

stating that "*deadlock* (`deadlock`) *never occurs* (G¬) *in any execution trace* (A)," is satisfied in a given system model.

In probabilistic model checking, probabilistic finite state machines are used as modelling language. The simplest probabilistic models are *Discrete-Time Markov Chains* (DTMCs), *Continuous-Time Markov Chains* (CTMCs), and *Markov Decision Processes* (MDPs). Formal properties are then expressed using *probabilistic logics*, a probabilistic extension of temporal logics. For discrete-time models (e.g., DTMCs and MDPs), the probabilistic extension of CTL, PCTL [13], is used and for continuous-time models (e.g., CTMCs) Continuous Stochastic Logic (CSL [14]) is used. Both languages can express quantitative expressions, such as *"the probability that deadlock never occurs is greater than 0.9,"* formally translated as

$$\mathrm{P}_{>0.9}\left[\mathrm{G}\neg\texttt{deadlock}\right]. \tag{2}$$

PCTL is defined according to the following grammar:

$$\begin{gathered}\varphi ::= \mathrm{true} \left|p\right| \neg\varphi \left|\varphi\wedge\varphi\right| \mathrm{P}_{\sim r}\left[\psi\right], \\ \psi ::= \mathrm{X}\varphi \left|\varphi \mathrm{U} \varphi\right| \varphi \mathrm{U}^{\mathrm{I}}\varphi,\end{gathered} \tag{3}$$

where p is a set of atomic propositions, $0 \le r \le 1$ is a *probability bound,* and $\sim\in \{<,>,\le,\ge,=\}$. The *probabilistic operator* $\mathrm{P}_{\sim r}$ is the probabilistic extension of branch/path quantifiers A and E of CTL. Informally speaking, $\mathrm{P}_{\sim r}[\psi]$ is satisfied at a state s if, and only if, the probability of taking the path from the state s which satisfies ψ meets the bound "$\sim r$." The *path* formulas ψ are constructed using the *next* (X) *until* (U) and *bounded-until* (U^{I}) operators. We can derive other temporal operators, such as F (*eventually*) and G (*always*) from X and U. For example, $\mathrm{F}\varphi \equiv \mathrm{true}\,\mathrm{U}\varphi$ and $\mathrm{G}\varphi \equiv \neg\mathrm{F}\neg\varphi$. The informal meanings of path formulas are as follows:

(i) $\mathrm{X}\varphi$ holds at a state on a path if, and only if, φ holds in the next state on the path;

(ii) $\mathrm{F}\varphi$ holds at a state on a path if, and only if, φ holds eventually at some future state on the path;

(iii) $\mathrm{G}\varphi$ holds at a state on a path if, and only if, φ holds at all future states on the path;

(iv) $\varphi_1\mathrm{U}\varphi_2$ holds at a state on a path if, and only if, φ_1 holds on the path up until φ_2 holds; and

(v) $\varphi_1\mathrm{U}^{\mathrm{I}}\varphi_2$ holds at a state on a path if, and only if, φ_2 holds on the path at some time step within the interval I and φ_1 holds at all preceding states on the path.

The formal semantics of the operators X, U, and U^{I} are defined as follows:

(i) $\mathscr{M},\sigma \vDash \mathrm{X}\varphi$ if and only if $\mathscr{M},\sigma[1] \vDash \varphi$;

(ii) $\mathscr{M},\sigma \vDash \varphi_1\mathrm{U}\varphi_2$ if and only if $\exists i \ge 0$ s.t. $\mathscr{M},\sigma[i] \vDash \varphi_2$ and (for all $j < i$) $\mathscr{M},\sigma[j] \vDash \varphi_1$;

(iii) $\mathscr{M},\sigma \vDash \varphi_1\mathrm{U}^{\mathrm{I}}\varphi_2$ if and only if $\exists i \in \mathrm{I} = [d,d']$ s.t. $\mathscr{M},\sigma[i] \vDash \varphi_2$ and (for all $d \le j < i$) $\mathscr{M},\sigma[j] \vDash \varphi_1$,

where $\mathscr{M}$ is a Markov chain (or an Markov decision process) and σ is a *path*, that is, a (possible infinite) sequences of states. The ith element of a path σ is denoted by $\sigma[i]$.

In addition to the operators of PCTL, the logic CSL also contains the *steady-state* (*long-run*) $\mathrm{S}_{\sim r}$ operator. Namely, the formula $\mathrm{S}_{\sim r}[\psi]$ holds in a state s if, and only if, the steady-state

```
module module_name

    //variable declarations
    s : [0...10] init 0;

    //state transitions
    []s = 0 → 0.4 : (s' = 1) + 0.6 : (s' = 2);

endmodule
```

Algorithm 1

probability of being in a state which satisfies ψ is bounded by "$\sim r$".

A probabilistic model checker can be used to verify if a PCTL or CSL formula—depending on the time semantics—holds in a given probabilistic model. In this paper, we particularly consider the model checkers Prism, Mrmc, and Ymer because of their support for these languages and useful features which they employ.

2.1. Prism.

Prism is the most widely used probabilistic model checking tool. It provides formal verification for different probabilistic models such as DTMCs, CTMCs, and MDPs, which are coded as the Prism's *reactive modules*, a simple state-based language. Prism is a *symbolic* model checker; namely, it uses a compact and structured representation of "data structures based on binary decision diagrams (BDDs) and multiterminal binary decision diagrams (MTBDDs)" in order to reduce the size of probabilistic models [15].

The Prism modelling language can be described as follows (summarised from [16]). The language is composed of a set of components, called *modules*, representing an encapsulation for different parts of the system and *variables* representing the state of each module. The global state of the model at any time point is determined by the values of all variables (and optionally global variables). Variables can be declared using the syntax

$$s : [0 \cdots 10]\ \mathbf{init} 0; \tag{4}$$

meaning that s is an integer variable whose values range between 0 and 10; that is, 0 is the *lower bound* and 10 is the *upper bound*. State transitions can be modeled using the following syntax:

$$[\text{act}]\ \text{guard} \longrightarrow \text{rate} : \text{update}, \tag{5}$$

where "act is an (optional) label, guard is a predicate over the variables of the model, rate is a (nonnegative) real-valued expression, and update is the values of variables in the next state" [15]. For example,

$$[\,]\ s = 0 \longrightarrow 0.4 : \left(s' = 1\right) + 0.6 : \left(s' = 2\right); \tag{6}$$

states that if the variable s is 0 then s will be 1 in the next state with probability 0.4 and will be 2 with probability 0.6, otherwise. A module is then described as shown in Algorithm 1.

The property specification languages supported by the tool include PCTL and CSL. In addition to probabilistic properties, Prism also supports *rewards* structures, providing reward properties based on quantitative information such as "expected number of deadlocks" and "expected time taken to reach a state where a deadlock occurs." This is done using "R" operator. Prism provides the following reward types: the *reachability* reward ($R_{\sim r}[F\varphi]$), *cumulative* reward ($R_{\sim r}[C \le t]$), *instantaneous* reward ($R_{\sim r}[I = t]$), and *steady-state* reward ($R_{\sim r}[S]$). These formulas can be intuitively described as follows [15]:

(i) $R_{\sim r}[F\varphi]$ asserts that the expected reward accumulated until φ is satisfied is bounded by $\sim r$;

(ii) $R_{\sim r}[C \le t]$ asserts that the expected reward accumulated until time t is bounded by $\sim r$;

(iii) $R_{\sim r}[I = t]$ asserts that the expected value of the state reward at time instant t is bounded by $\sim r$;

(iv) $R_{\sim r}[S]$ asserts that the long-run average expected reward is bounded by $\sim r$.

We remark that rewards are added to model files. The reward structures denote labellings of states and transitions with associated real values. These values are then used when calculating the accumulated or instantaneous rewards.

Prism provides both a command-line tool and a graphical user interface. The latter provides (i) a model editor for the description/modelling language; (ii) an editor for property specification langauge; (iii) a simulator tool to debug model execution traces; and (iv) graph-plotting tools [16]. Prism also provides an approximate/statistical model checking feature, but this is only allowed to a subset of PCTL/CSL formulas. For example, statistical model checking cannot be used in verification of steady-state properties.

2.2. Mrmc.

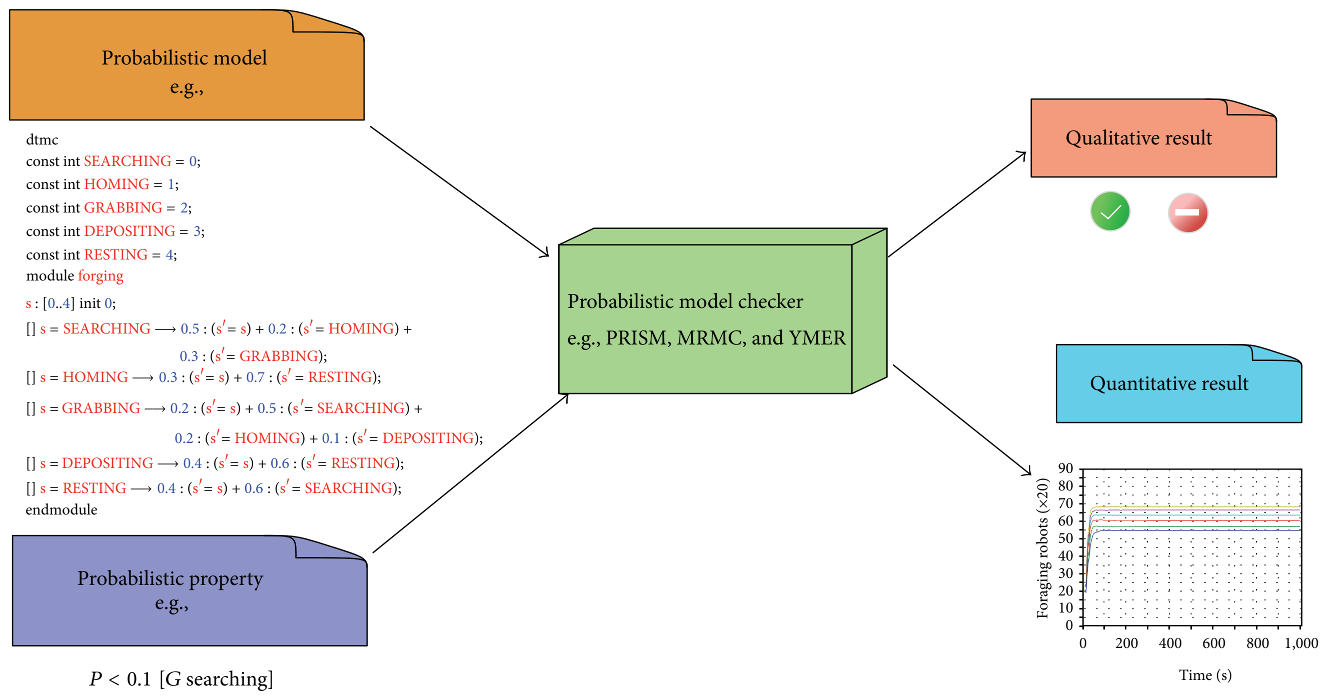
Mrmc is another probabilistic model checking tool, which allows formal verification for DTMCs and CTMCs (as well as Continuous-Time Markov Decision Processes, a nondeterministic variant of CTMCs). As in Prism, DTMCs and CTMCs are verified against PCTL and CSL formulas, respectively.

Mrmc also features the logics PRCTL and CRSL, which are an extension of PCTL and CSL, respectively, with rewards structures. The syntax of the reward formulas are slightly different than Prism's reward formulas [17]: *reachability* reward: $E[t][r_1, r_2][\varphi]$, *cumulative* reward: $Y[t][r_1, r_2][\varphi]$, *instantaneous* reward: $C[t][r_1, r_2][\varphi]$, and *steady-state* reward: $E[r_1, r_2][\varphi]$. The rewards represent expected rewards per time unit and are checked against a reward bound $[r_1, r_2]$. For example,

$$Y\,[t]\,[r_1, r_2]\,[\varphi] \tag{7}$$

meaning that the expected accumulated reward rate per time unit in φ states until the tth transition will be within the interval $[r_1, r_2]$.

Some important features of Mrmc are summarised as follows [9]. Unlike Prism, Mrmc is a command-line explicit



```
dtmc
const int SEARCHING = 0;
const int HOMING = 1;
const int GRABBING = 2;
const int DEPOSITING = 3;
const int RESTING = 4;
module forging
s : [0..4] init 0;
[] s = SEARCHING ⟶ 0.5 : (s′ = s) + 0.2 : (s′ = HOMING) +
              0.3 : (s′ = GRABBING);
[] s = HOMING ⟶ 0.3 : (s′ = s) + 0.7 : (s′ = RESTING);
[] s = GRABBING ⟶ 0.2 : (s′ = s) + 0.5 : (s′ = SEARCHING) +
              0.2 : (s′ = HOMING) + 0.1 : (s′ = DEPOSITING);
[] s = DEPOSITING ⟶ 0.4 : (s′ = s) + 0.6 : (s′ = RESTING);
[] s = RESTING ⟶ 0.4 : (s′ = s) + 0.6 : (s′ = SEARCHING);
endmodule
```

Figure 1: Standard approach for probabilistic model checking.

state model checker employing a "numerical solution engine." It has been therefore used as a back-end model checking tools for various systems including Petri nets, process algebras, and stochastic hybrid systems. Mrmc has recently been improved with a better memory management and implementation of the sparse matrices and support for bisimulation minimization. This increases its efficiency and performance on large models.

Mrmc also employs a "discrete-event simulation engine" for statistical CSL model checking. In statistical model checking, rather than exhaustively exploring the entire state space, the system is simulated finitely many times to obtain execution traces and statistical evidence is provided for the verification of a property [18]. Unlike Prism, Mrmc's statistical model checking covers the entire formula set of CSL, including steady-state properties.

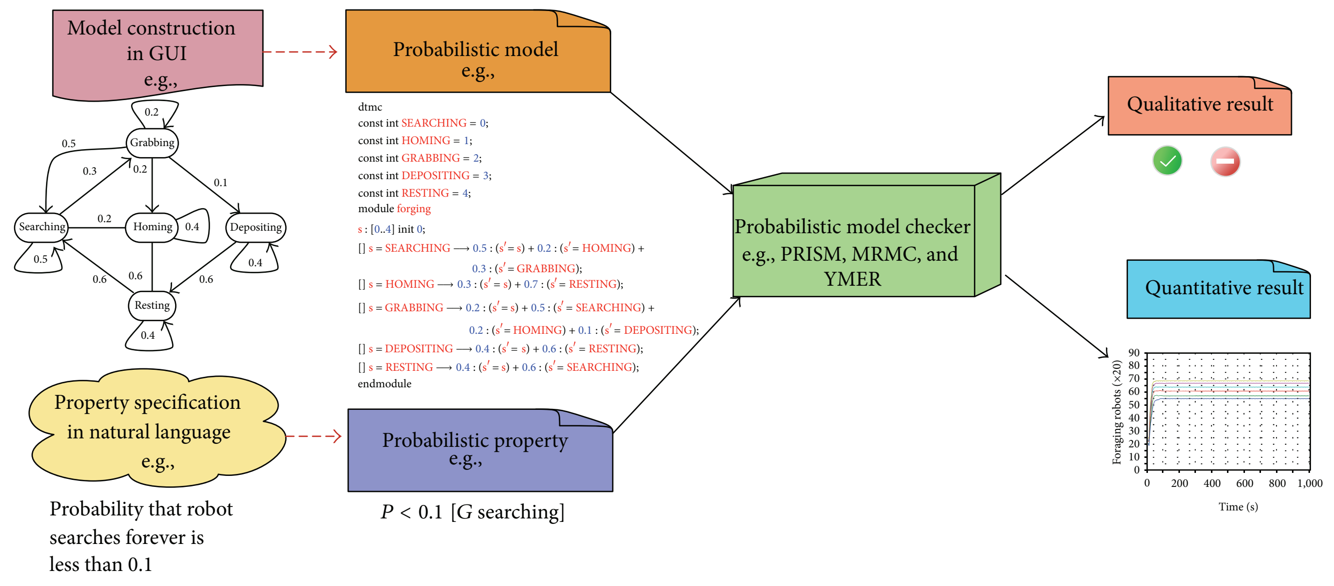
2.3. Ymer. Ymer is a *statistical* model checking tool, used to verify transient properties of CTMCs [10]. So, unlike Prism and Mrmc, it does not exhaustively analyse all system behaviour. Instead, it employs statistical techniques relying on "discrete event simulation and sequential acceptance sampling" [10]. The modelling language of Ymer is very similar to that of Prism with a difference that Ymer also supports *generalized semi-Markov processes*. The property specification language of the tool is CSL, but Prism and Mrmc support a richer set of properties than Ymer.

3. Light-Weight Approach to Probabilistic Model Checking

As in classical model checking, a probabilistic model checker requires two inputs: (i) a probabilistic model of the system to be analysed and (ii) a probabilistic property. As discussed above, a probabilistic model can be a DTMC, CTMC, or MDP, which is coded according to the high-level modelling language of the model checker, for example, reactive modules in case of Prism. A probabilistic property is the formal specification of a requirement to be checked, which is expressed in one of the probabilistic logics described above, for example, PCTL when the system is modeled in DTMC. The model checker then automatically checks if the model satisfies the given specification. Based on the type of the property, it produces either a qualitative answer (a "yes" or "no") or a quantitative result. If the property is not satisfied, the model checker also produces a counterexample to help the modellers debug the output and find the bug. The overall process is illustrated in Figure 1.

In standard approach, models are created manually using a high-level modelling language tailored to a particular model checker and properties are specified in a specific formal logic/language. This, however, requires having a good understanding of both modeling and property specification and being familiar with the syntax of both high-level description and specification languages. Unfortunately, this is not an easy task for nonexperts to learn description languages for modelling and formal logics for property specification. In particular, property specification is very daunting and error-prone for nonexperts and even for experts in case of some complex properties.

To make this process easier for nonexperts, we propose an approach facilitating probabilistic model checking by providing an abstraction over high-level modeling and property specification languages. Namely, we suggest creating a system model using a *graphical user interface* and generate


Model construction in GUI e.g.,
Probabilistic model e.g.,
Property specification in natural language e.g.,
Probability that robot searches forever is less than 0.1
Probabilistic property e.g.,
P < 0.1 [G searching]
Probabilistic model checker e.g., PRISM, MRMC, and YMER
Qualitative result
Quantitative result
Searching
Grabbing
Homing
Depositing
Resting
Foraging robots (×20)
Time (s)

```
dtmc
const int SEARCHING = 0;
const int HOMING = 1;
const int GRABBING = 2;
const int DEPOSITING = 3;
const int RESTING = 4;
module forging
s : [0..4] init 0;
[] s = SEARCHING ⟶ 0.5 : (s′ = s) + 0.2 : (s′ = HOMING) +
                   0.3 : (s′ = GRABBING);
[] s = HOMING ⟶ 0.3 : (s′ = s) + 0.7 : (s′ = RESTING);
[] s = GRABBING ⟶ 0.2 : (s′ = s) + 0.5 : (s′ = SEARCHING) +
                  0.2 : (s′ = HOMING) + 0.1 : (s′ = DEPOSITING);
[] s = DEPOSITING ⟶ 0.4 : (s′ = s) + 0.6 : (s′ = RESTING);
[] s = RESTING ⟶ 0.4 : (s′ = s) + 0.6 : (s′ = SEARCHING);
endmodule
```

FIGURE 2: A light-weight approach for probabilistic model checking.

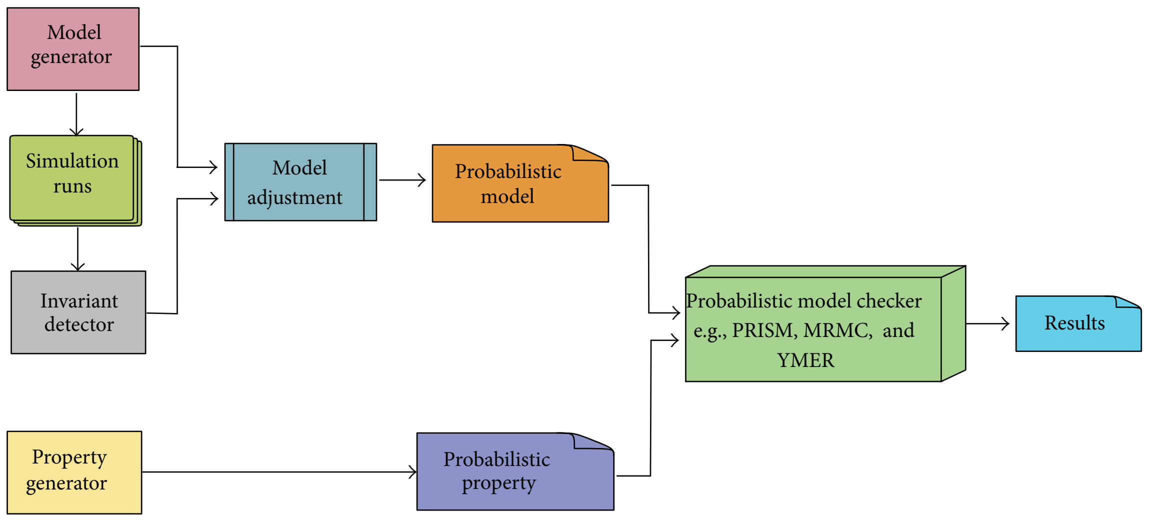


FIGURE 3: System overview.

informal properties using *natural language* statements. The constructed model is then translated to the corresponding high-level modelling language, and the informal properties are translated into their formal counterparts. Our approach is illustrated in Figure 2. Here, we consider the models constructed as probabilistic state machines and natural language statements as probabilistic properties, because in this paper we focus on probabilistic model checking.

The system overview of our approach is shown in Figure 3, summarised as follows.

Model Construction. A system model is constructed using the *Model Generator*, which is a graphical user interface providing drawing features to construct probabilistic state machines. The constructed model is then translated into a high-level modelling language, which then becomes an input to the corresponding model checker.

Reducing Model Size. A well-known problem with model checking is the *state explosion* problem. Namely, the number of states increases so rapidly that model checking cannot become feasible any more. The size of model, mostly depending on the number of states, is an important factor for the model checking efficiency, because the resources required to perform model checking are very sensitive to the model size. Unfortunately, it is very easy to create huge models when variables are unnecessarily assigned to very large ranges. A good strategy to tackle the state explosion problem and increase the performance of model checking is therefore to reduce the number of states. As discussed in Section 2.1, variable ranges are defined using a *lower* and an *upper* bound. In most cases, users might not be able to know the exact lower and upper bounds. If the estimation of variable ranges is not realistic, then the resulting model size will be very large.

In order to tackle this issue, we devise a method automatically estimating a reasonable lower and upper bound for model variables. In order to do this, we simulate the model generated by the Model Generator, use an *invariant detector* to analyse the simulation results, and acquire the lower and upper bounds for variables. In the *Model Adjusment* process we then update the model variables with the bounds acquired from the invariant detector. In this way, even if the users provide very large bounds, we can reduce them and update the model accordingly. The updated model is then given to the model checker as an input.

Property Construction. The *Property Generator* tool provides a graphical user interface allowing users to create properties easily by manipulating a configurable form with some interfaces such as combo and text boxes. So, by only selecting natural language statements, users can generate a set of informal properties which are automatically translated into their formal counterparts. The formal properties can be directly given to the corresponding model checker as an input.

4. Model Construction

As part of the proposed methodology, we aim to use various tools to make the modeling task less complicated for nonexperts. In this section, we provide a more detailed account for the tools and methods used.

4.1. Model Generator.

The *Model Generator* component relies on the prototype front-end tool, called *Drawing* Prism, developed in [19]. The Drawing Prism tool, DP in short, provides drawing features to construct probabilistic state machines, which are then translated into a Prism model file.

DP implements two interfaces: (i) *drawing interface* allowing users to draw probabilistic state machines—see Figure 4—and to populate their state machines as a data model; (ii) *translator*, which translates the data model representing the probabilistic state machine into the Prism's high-level modelling language.

DP employs ArgoUML [20] as the drawing tool. ArgoUML is an open source project, and it supports most popular standards, for example, UML, XMI, and SVG. It is written in Java; it is therefore supported by any platform with a Java environment.

ArgoUML converts a state diagram into an XMI data. XMI, standing for XML-based Metadata Interchange Format, is mainly used as a model interchange format for UML. Although XMI is not an easy format to read, it has some advantages, for instance, its abstract tree representation which makes parsing easy. XMI stores all the necessary information in a state machine, for example, states, transitions, labels, and probability values. The translator extracts all

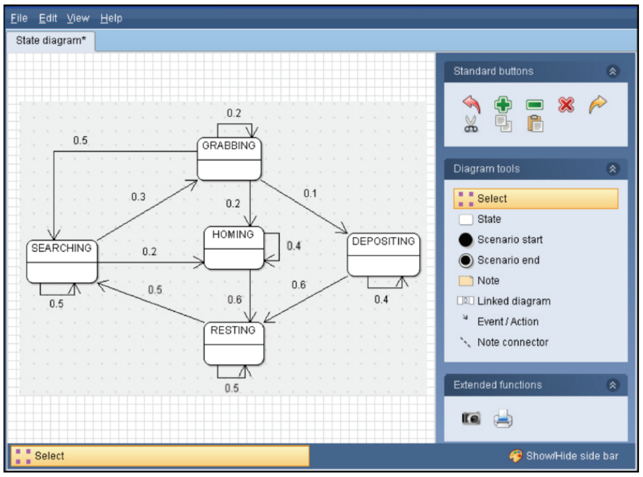


Figure 4: A probabilistic state machine constructed in DP [19].

information from a state machine in XMI and then translates it into the format accepted by Prism.

The DP function design is presented in Figure 5. The functional components can be summarised as follows [19].

(i) *User Interface*. "It has a control and drawing panel through which users access DP and interact closely with system operators. Control components provide a common control surface used by GUI, scripts, and programmatic access. The Interface is used to integrate the rest of these subsystems."

(ii) *Drawing System*. It classifies every component of the state machine drawn using the user interface and then integrates these components in the data structure.

(iii) *XMI Data*. It is the data structure acquired from the Drawing System and translated to the XMI format.

(iv) *Translator*. It translates state machine information in the XMI data structure to a Prism model file using specific algorithms.

4.2. Invariant Detector.

The *Invariant Detector* component generates reasonable lower and upper bounds for model variables by automatically detecting invariants through analysing the simulation traces produced by *Model Generator*. For this task we employ the Daikon tool [21].

Daikon is an invariant detector tool which dynamically reports invariants in a program. An *invariant* is simply a mathematical property, such as $x = 3$, $y \leq 2x$, z is one of $\{0, 1\}$, holding at specific execution points of a program. Daikon executes a code, analyses the values of program variables, and then detects invariants which are true over certain points.

Daikon is originally developed to support high-level programming languages, for example, C, C++, Java, and Perl. However, its source code is freely available and can be modified to be able to use it in other formats. In order to facilitate its usage in detecting invariants from simulation traces, we have extended its functionality. After analysing the simulations, it produces some mathematical relations

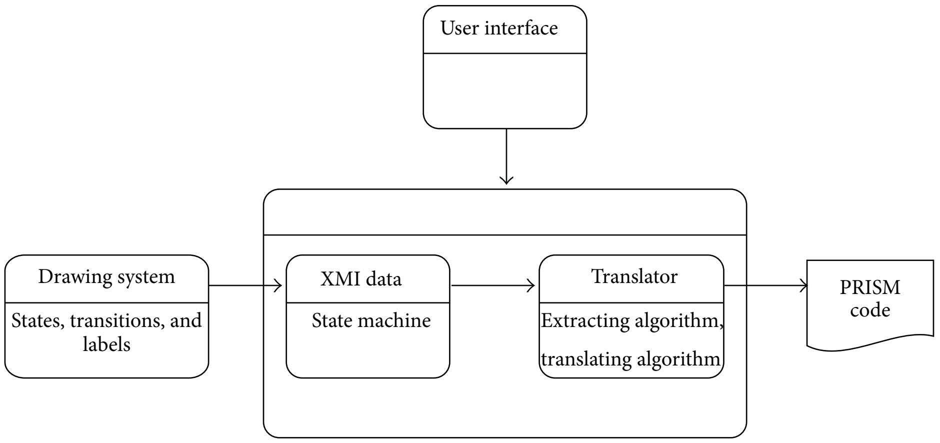
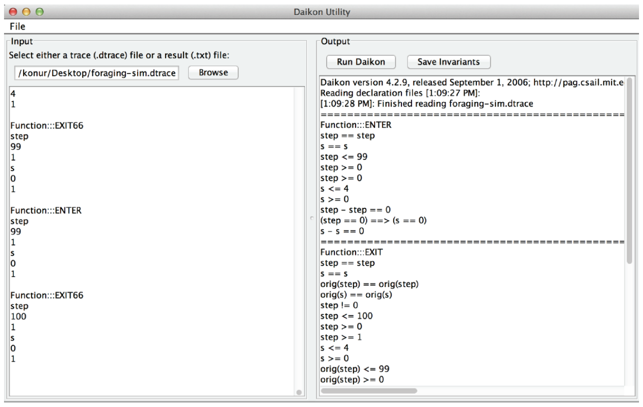


Figure 5: Functional components of DP [19].


Daikon Utility
File
Input
Select either a trace (.dtrace) file or a result (.txt) file:
Browse
Output
Run Daikon
Save Invariants

```
/konur/Desktop/foraging-sim.dtrace
```

```
4
1

Function:::EXIT66
step
99
1
s
0
1

Function:::ENTER
step
99
1
s
0
1

Function:::EXIT66
step
100
1
s
0
1
```

```
Daikon version 4.2.9, released September 1, 2006; http://pag.csail.mit.e
Reading declaration files [1:09:27 PM]:
[1:09:28 PM]: Finished reading foraging-sim.dtrace
=================================================
Function:::ENTER
step == step
s == s
step <= 99
step >= 0
step >= 0
s <= 4
s >= 0
step - step == 0
(step == 0) ==> (s == 0)
s - s == 0
=================================================
Function:::EXIT
step == step
s == s
orig(step) == orig(step)
orig(s) == orig(s)
step != 0
step <= 100
step >= 0
step >= 1
s <= 4
s >= 0
orig(step) <= 99
orig(step) >= 0
```

Figure 6: Daikon invariant detector.

between various model variables. The ranges provided by Daikon then become a realistic lower and upper bound.

Daikon was previously used to detect invariants to construct formal specifications [22, 23]. In contrast, here we use the invariants to estimate reasonable bounds for model variables, as discussed above. In [24], Daikon was used to formulate temporal logic formulas to be model checked by NuSMV. Here, we extend the work done in [24] by adjusting it to accept the outputs of Prism's discrete event simulator. A screen shot of the tool is shown in Figure 6.

Depending on the quality of information contained in execution traces, Daikon can report many types of invariants, such as *arithmetic* ($y \le 2x$), *non-zero* (e.g., $x \ne 0$), *element of* (e.g., x is one of $\{0, 1\}$), and *interval* (e.g., $0 \le x \le 5$). Daikon might also report some redundant invariants (e.g., $x == x$). However, it employs a filtering mechanism allowing to omit redundant and unwanted invariants to be returned. In this way, we can only obtain *upper bound* (e.g., $x \le 2$) and *lower bound* (e.g., $x \ge 0$) invariants.

Remark. The Daikon tool is originally developed to detect invariants within a program written in a high-level programming language. In a typical programme, there can be hundreds of variables. Daikon is capable of reporting invariants for such large numbers of variables over thousands of executions. A typical system for which model checking is feasible probably contains tens of model variables in the model description (which might lead to millions/billions of states/transitions when the model is constructed by the model checker). It will be sufficient to obtain around a hundred simulation traces to find the bounds for model variables. This suggests that the resources required to use the Daikon tool to obtain variable bounds are not more than those required by its original usage.

4.3. Model Adjustment. In the *Model Adjustment* process we perform two tasks as follows. (i) We update the bounds of model variables in the Prism model file with the bounds acquired from the invariant detector. In this way, even if the users provide very large bounds, we can reduce them, and update the model accordingly. The updated model is then given to Prism as an input. (ii) We also translate the same model to the corresponding Mrmc language. At the moment, we are using Prism's built-in export facility to translate Prism models to the Mrmc format. The modelling language of Ymer is very similar to that of Prism, which requires a few minor changes. Currently, we make these changes manually, but an automatic translation is a very primitive process.

5. Property Construction

Property specification is very daunting and error-prone for nonexperts. It is even cumbersome for experts to specify complex properties. In most cases, properties are not intuitive and self-explanatory, so their meanings will not be clear to those not familiar with formal methods. This unfortunately hinders reusability of properties already studied in the literature and makes them inaccessible to a wide audience. Another issue that we would like to address is that although a formal property can be syntactically correct it might not be a valid representation of the requirement we wish to verify [12]. This is especially the case when the formal specification is complex and long.

In order to tackle these issues and facilitate property specification, we propose the *Property Generator* tool, which provides guidance to construct properties using the *natural* *language statements* representing informal properties and then automatically translates them to their formal counterparts. Users are able to construct properties easily by manipulating a configurable form with some interfaces such as combo and text boxes, which makes the property specification a very simple and intuitive task.

The Property Generator tool features a set of *property patterns* based on most recurring properties studied in the literature. These patterns provide a systematical classification, guiding users to construct natural language expressions representing the properties that they want to express. The tool works based on a *structured grammar* defined for both natural language representation of these patterns and their formal translations.

The idea of categorising properties into a set of patters initially started in [12], where several hundreds of properties were analysed. Since this seminal work, there have been several studies in this direction. [25] extended the pattern classes of [12] with more time related patterns and their "associated observer automata." [26] provided a set of patterns in the context of real-time specification. [27] introduced a unified pattern system extended with a new set of real-time pattern classes. [28] analysed probabilistic properties and provided a probabilistic category system along with a structured grammar. In [29] we proposed a set of query templates similar to the one presented in this paper which targeted biological models.

A subset of the patterns which we used in the Property Generator tool and their formal translations in Prism and Mrmc are shown in Table 1. In the table, *phi* and *psi* are state expressions returning true or false (e.g., $x \leq 4$), $\bowtie \; \in \{<, >, \leq, \geq\}$ is a relational operator, $p \in [0, 1]$ is a real, $r, r1, r2$ are real numbers, and $t, t1, t2$ are integers. Here, where applicable, we provide both the *unbounded* and *bounded* versions of the patterns. Note that Ymer can express only a subset of these patterns, and we therefore do not include it in the table.

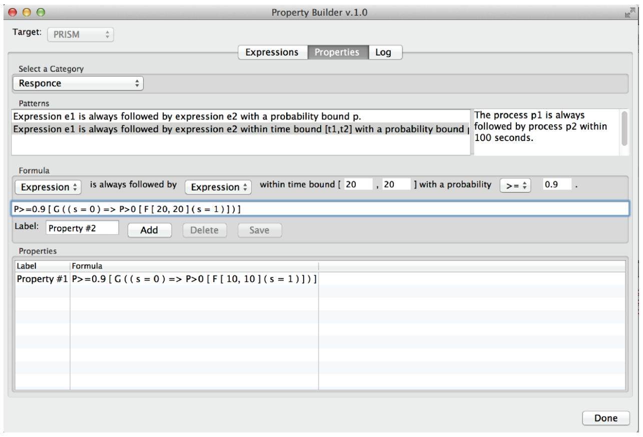
As Table 1 illustrates, properties in natural language are much more intuitive and comprehensive than their formal translations. For example, the natural language representation of the *Bounded Response* pattern

$$\text{“}\textit{phi} \text{ is always followed by } \textit{psi} \text{ within time bound } [t1, t2] \text{ with a probability } \bowtie p\text{”} \tag{8}$$

is much more easier than its formal translation, for example, Mrmc,

$$\texttt{P}\,\{\bowtie\texttt{p}\}\,[\,!\,(\texttt{tt U }!(\texttt{phi => P >= 1}\,[\texttt{tt U}\,[\texttt{t1,t2}]\,\texttt{psi}]))]\,. \tag{9}$$

The Property Generator tool makes the property construction a very easy and effortless task without requiring the knowledge about the formal syntax of the target model checker. For example, to build this property, the user selects the *bounded* version of the *Response* pattern using the graphical user interface provided. The user only needs to provide the values for *phi*, *psi*, $t1$, $t2$, $\bowtie$, and p. The formal translation is then done automatically.



Target: PRISM

Select a Category: Responce

Patterns:

Expression e1 is always followed by expression e2 with a probability bound p.

Expression e1 is always followed by expression e2 within time bound [t1,t2] with a probability bound

The process p1 is always followed by process p2 within 100 seconds.

Formula: Expression is always followed by Expression within time bound [20 , 20] with a probability >= 0.9 .

```
P>=0.9 [ G ( ( s = 0 ) => P>0 [ F [ 20, 20 ] ( s = 1 ) ] ) ]
```

Label: Property #2

Properties

Label	Formula
Property #1	P>=0.9 [G ((s = 0) => P>0 [F [10, 10] (s = 1)])]

Figure 7: Property Generator tool.

A screen shot of the Property Generator tool is illustrated in Figure 7. We developed a previous version of the tool to construct (linear) temporal properties used in formal verification of *kP System* models [30].

The operation of the tool during a property construction process is as follows. The user first selects a target, for example, Prism, Mrmc, and Ymer, to which the property is translated. When a pattern is selected from the category combo box, a template (for bounded and unbounded cases) is displayed in the text field. When the appropriate template is clicked, both the informal representation and its formal translation appear in the corresponding text box. Also, a typical example of the selected pattern is displayed at another text field. In order to complete the formula, the user is required to select and fill in the missing values. The selection and typing are interactively shown in the text box displaying the formal translation. When the formula is complete, the user can save it. If there are still missing fields to be completed, he/she receives a warning. The user can edit a saved property at any time.

The main components of the Property Generator tool are shown in Figure 8. The *Expression Builder* component of the *Property Generator GUI* constructs atomic expressions, that is, state formulas returning *true* or *false*, using model variables and constants, and Boolean expressions using atomic expressions. The grammar for building expression is defined in the *Expressions* file. The *Property Builder* of the GUI constructs properties using the generated expressions and a set of patterns whose grammar is defined in the *Patterns* file. The *Cached Data* component keeps the expressions and properties created or managed throughout a user session. It works as a repository to keep the generated data until they are saved to a file and the application closes.

Remark. The performance of the Property Generator tool does not depend on the model size. The properties are constructed from a set of patterns. Once the user fills in the empty fields, the construction is done instantly. So, it does not matter whether one uses this tool for a small example or a large example.

Table 1: Property patterns and their PRISM and MRMC translations.

Property in natural language	PRISM translation	MRMC translation
phi will eventually hold, until then *psi* holds with a probability ⋈*p*	`P⋈p [psi U phi]`	`P{⋈p} [psi U phi]`
phi will hold within time bound [*t*1, *t*2], until then *psi* holds with a probability ⋈*p*	`P⋈p [psi U[t1, t2] phi]`	`P{⋈p} [psi U[t1, t2] phi]`
phi always holds with a probability ⋈*p*	`P⋈p [G phi]`	`P{⋈p} [!(tt U !(phi))]`
phi always holds within time bound [*t*1, *t*2] with a probability ⋈*p*	`P⋈p [G[t1, t2] phi]`	`P{⋈p} [!(tt U[t1, t2]!(phi))]`
phi is always followed by *psi* with a probability ⋈*p*	`P⋈p [G (phi => P >= 1 [true U psi])]`	`P{⋈p} [!(tt U !(phi => P >= 1 [tt U psi]))]`
phi is always followed by *psi* within time bound [*t*1, *t*2] with a probability ⋈*p*	`P⋈p [G (phi => P >= 1 [true U[t1, t2] psi])]`	`P{⋈p} [!(tt U !(phi => P >= 1 [tt U[t1, t2] psi]))]`
phi holds infinitely often with a probability ⋈*p*	`P⋈p [G (P >= 1 [true U phi])]`	`P{⋈p} [!(tt U !(P >= 1 [tt U phi]))]`
The expected accumulated reward until *phi* holds is ⋈*r*	`R⋈r [true U phi]`	
The expected accumulated reward until *phi* holds within *t* time units is between *r*1 and *r*2		`E[t][r1, r2][phi]`

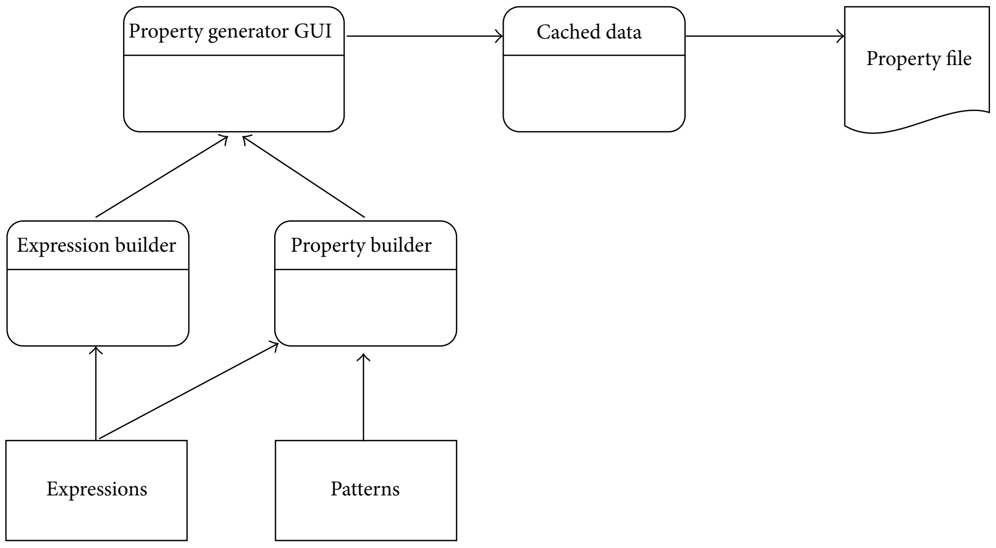


Figure 8: Main components of the Property Generator tool.

6. An Example: Foraging Robots

In this section, we illustrate our approach on an example system. Here, we choose the *foraging robot scenario*, presented in [31]. We analysed the system in [32, 33], which can be described as follows:

> "Within a fixed size arena, there are a number of foraging robots; that is, each robot must search a finite area and bring food items back to the common nest. Food is placed randomly over the arena and more may appear over time. There is no guarantee that robots will actually find any food.
>
> The behaviour of each robot in the system is represented by the probabilistic state machine in Figure 9, comprising the states: (i) SEARCHING, wherein the robot is searching for food items; (ii) GRABBING, wherein the robot attempts to grab a food item it has found; (iii) DEPOSITING, wherein the robot moves home with the food item; (iv) HOMING, wherein the robot moves home without having found food; and (v) RESTING, wherein the robot rests for a particular time interval."

In this scenario, we assume that all transitions occur with a probability. The state machine in Figure 9 operates on the following probabilities: γ_f (the probability of finding a food item), γ_g (the probability of grabbing a food item), γ_h (the probability of moving to HOMING state), γ_r (the probability of moving to RESTING state), and γ_s (the probability of moving to SEARCHING state).

We can draw this machine using the Drawing PRISM tool as in Figure 4. Here we assume $\gamma_f = 0.3$, $\gamma_g = 0.1$, $\gamma_h = 0.2$, $\gamma_r = 0.6$, and $\gamma_s = 0.5$. Note that this state machine represents the behaviour of an individual robot; however, a robot swarm is a collection of (often) identical robots working together. We therefore need a set of state machines to model the dynamic behaviour of the overall swarm. Fortunately, the Drawing PRISM tool has a feature allowing us to have a multiple copies of a probabilistic model, working as a parallel composition. Using this feature we can model the overall swarm.

The DP tool automatically translates the probabilistic state machine in Figure 4 to the PRISM's high-level modelling language, given in Algorithm 2.

After obtaining the model code, we simulate the model and have a number of simulation traces. We then run the invariant detector tool over these traces to estimate a lower and upper bound for the model variable `s`. Not surprisingly, we obtain the following constraints:

$$\mathtt{s} >= 0, \qquad \mathtt{s} <= 4. \tag{10}$$

The model therefore will not be adjusted. We remark that, since the model is very intuitive, we could give the exact range of the variable `s` because we know the values thats will take. Daikon has then returned the precise same bounds. However, in realistic cases, users might not be able to know the exact lower and upper bounds, and range estimations might not be realistic. The invariant detector will then provide more realistic bounds and reduce the state space.

We now construct some properties using the Property Generator tool and property patterns provided. Our strategy to build properties is as follows:

(i) first specify the properties that we are interested informally;

(ii) construct the corresponding natural language representations using the Property Generator tool;

(iii) select the target language to which the properties are converted automatically;

(iv) run model checking experiments using the translated formal properties.

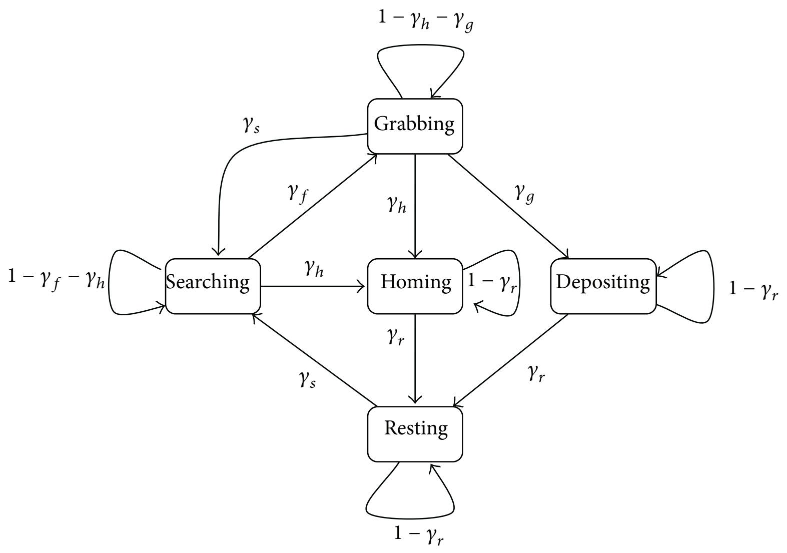


FIGURE 9: Probabilistic state machine for a foraging robot.

```
dtmc

const int SEARCHING = 0;
const int HOMING = 1;
const int GRABBING = 2;
const int DEPOSITING = 3;
const int RESTING = 4;

module foraging
  s : [0...4] init 0;
[] s = SEARCHING -> 0.5 : (s' = s) + 0.2 : (s' = HOMING) + 0.3 : (s' = GRABBING);
[] s = HOMING -> 0.4 : (s' = s) + 0.6 : (s' = RESTING);
[] s = GRABBING -> 0.2 : (s' = s) + 0.5 : (s' = SEARCHING) + 0.2 : (s' = HOMING)
     + 0.1 : (s' = DEPOSITING);
[] s = DEPOSITING -> 0.4 : (s' = s) + 0.6 : (s' = RESTING);
[] s = RESTING -> 0.5 : (s' = s) + 0.5 : (s' = SEARCHING);
endmodule
```

ALGORITHM 2

Table 2 shows informal, natural language and formal specifications of some properties. Here, we translate the properties to the PRISM's property language. The results of the verification experiments are also illustrated in the table. We remark that a property can be constructed as a query using ? symbol. In this case, PRISM returns the corresponding probability result after verifying the query. In Table 2, we show some query samples.

Multiple Robots. Figure 9 corresponds to a single robot behaviour, comprising five states that the robot visits during the execution of the system. However, a foraging swarm contains several robots. To model such a swarm, we can construct a state machine for each robot in the swarm and then take the *product of* all these to provide the behaviour of the overall swarm. To calculate the product, we take a simple and synchronous view. Namely, for each state machine A, say $A_1, A_2, A_3, \ldots$, and the state machine B, say $B_1, B_2, B_3, \ldots$, given that the transitions $A_1 \rightarrow A_2$ are labelled by α and $B_1 \rightarrow B_2$ are labelled by β, we can have a transition $A_1B_1 \rightarrow A_2B_2$ labelled by $\alpha\beta$ where A_1B_1, A_2B_2, and so forth are states in the product state machine and $\alpha\beta$ is a consistent label. This is done for every possible pair of transitions. The overall swarm system as the product of individual robots is shown in Figure 10. The DP tool permits creating multiple instances of modules, allowing composing individual state machines.

Table 3 compares the state spaces for different swarm populations. The table shows that the model size exponentially grows when the swarm size increases. We note that the construction times illustrated in Table 3 correspond to the building models by the model checker. The translation of models into the model checking language is not significantly affected by the size of the model. Namely, the time to translate

TABLE 2: Sample properties for a single robot.

Property	Informal, Natural language and Prism specifications	Result
1	*The robot eventually grabs food with a probability greater than 0.9* `s = 3` will eventually hold, until then `true` holds with a probability >0.9 `P > 0.9 [true U s = 3]`	TRUE
2	*What is the probability that the robot grabs food within 50 s?* `s = 3` will eventually hold within time bound [0, 50], until then `true` holds with a probability? `P = ? [true U[0, 50] s = 3]`	0.55
3	*What is the probability that the robot searches for food for 50 s. before going to home?* `s = 1` will hold within time bound [0, 50], until then $s = 0$ holds with a probability? `P = ? [s = 0 U[0, 50] s = 1]`	0.39
4	*The robot never goes to home within 50 s* `s! = 1` always holds within time bound [0, 50] with a probability >=1 `P >= 1 [G[0, 50] s! = 1]`	FALSE
5	*The robot does not continuously search for food forever* `s = 0` always holds with a probability <=0 `P <= 0 [G s = 0]`	TRUE
6	*When searching starts, the robot eventually grabs food with a probability greater than 0.6* `s = 0` is always followed by `s = 3` with a probability >0.6 `P > 0.6 [G (s = 0 => P >= 1 [true U s = 3])]`	TRUE
7	*The robot repeats its behaviour forever (e.g. searches for food)* `s = 0` holds infinitely often with a probability >=1 `P >= 1 [G (P >= 1 [true U s = 0])]`	TRUE

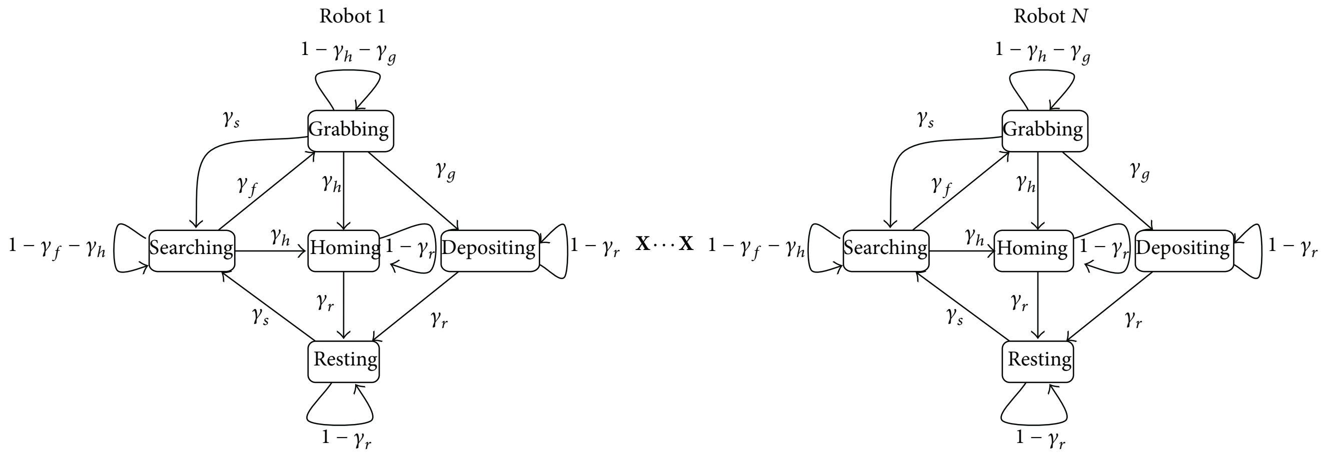


FIGURE 10: Probabilistic state machine for a foraging swarm.

TABLE 3: State space for different swarm populations.

Number of robots	States	Transitions	Model construction
1	5	13	0.001 seconds
2	25	105	0.002 seconds
3	125	725	0.003 seconds
4	625	4625	0.006 seconds
5	3125	28125	0.008 seconds
10	9765625	166015625	0.024 seconds
20	95367431640625	3147125244140625	0.733 seconds

a model into the model checking format can be neglected with respect to the resources required for model checking.

7. Discussion

Standard model checking methods work on a basic principle: a model is described in a certain modelling paradigm, for example, a probabilistic model; a property is specified in a certain logic, for example, a probabilistic temporal logic; and a suitable model checker is then used, for example, a probabilistic model checker. This approach suggests that whatever model checker the user selects, he/she has to analyse his/her system based on the aspects of modelling and specification languages. If he/she wants to use a different model checker, the user must start all over again. This, of course, means learning the syntax of all model checkers to be used, which is surely a big cumbersome for nonexperts (even for experts in some cases).

Another drawback is that existing specification languages are limited to a certain dimension, for example, time and probability. This unfortunately prevents specifying and verifying multidimensional behaviour, which can reveal more novel information about system behaviour.

We believe next generation model checkers should be able to do more than standard model checkers and hence should employ some new features in order to eliminate these drawbacks. Here, we discuss two of such features that we can integrate into our methodology.

Generic Model Checking. Although we have presented our approach for probabilistic model checking, it can be generalised to cover model checking for temporal, real-time, and probabilistic temporal logics. The property generator tool we have proposed is very feasible as it supports the translation of natural language statements to any logic. We can also extend the pattern list presented in Table 1 to cover temporal and real-time logics.

As we already know, the semantics of different logics are defined over different structures. For example, temporal, real-time, and probabilistic temporal logics are usually interpreted over Kripke structures, timed automata (TA) [34], and probabilistic models (e.g., DTMCs, CTMCs, and MDPs), respectively. We remind that a timed automaton models real-time behaviour using a finite set of real-valued *clocks* associated with states and transitions.

In order to allow model checking for these logics in the same platform, we therefore need a generic structure to cover all these models. Probabilistic timed automata (PTA) [35] can be used as a generic structure, because PTA extend TA with discrete probability distributions and embed Kripke structures, timed automata, and probabilistic models. We can construct PTA using the Drawing PRISM tool because it already supports labelling the states and transitions. If we allow clock constraints in labels, we can obtain PTA, which can then be instantiated to construct corresponding models. Namely, to construct a probabilistic model we disable clocks (i.e., state and transition labels do not include clock constraints); to construct a timed automaton we disable probability distributions (i.e., transition labels do not include probabilities); and to contract Kripke structures we disable both clocks and probability distributions. Models constructed using the tool can then be translated to the modelling languages of corresponding model checkers. For example, Kripke structures are translated to the PROMELA language for SPIN, timed automata are translated to the UPPAAL's description language, and probabilistic state machines and probabilistic timed automate are translated to the PRISM's reactive modules.

This general approach will allow us to use model checkers for temporal logics, for example, SPIN and NUSMV, real-time model checkers, for example, UPPAAL and KRONOS, and probabilistic model checkers, for example, PRISM and MRMC, in the same platform.

Combined Model Checking. Standard model checking is defined for standalone logics. We can also develop a model checking strategy for *combined logics*. A combined logic synthesizes different aspects using constituent logics, for example, (i) classic temporal logics (CTL, LTL, etc.); (ii) belief/knowledge logics (modal logics KD45, S5, etc.); (iii) probabilistic temporal logics (PCTL, etc.); and (iv) real-time temporal logics (TCTL, etc.). We can then write statements specifying multidimensional behaviour. For example, assume that, in our foraging scenario, we want to combine the belief and probability aspects, which can be done by combining PCTL and KD45. The statement

> *The robot i believes that the probability of robot j's grabbing a food item within 50 s. is greater than 0.9*

is expressed in the combined logic as

$$\mathrm{B}_i\left[\mathrm{P}_{>0.9}\left(\text{true}\ \mathrm{U}^{[0,50]} s_j = 3\right)\right] \tag{11}$$

which cannot be stated using standard logics.

Based on the combination strategy, a model checking procedure can be defined. In [36] we provide a generic model checking method, which synthesizes a combined model checker from the model checkers of simpler constituent logics. We also "show that the complexity of the synthesized model checker is essentially the supremum of the complexities of the component model checkers" [36].

This result suggests that we can devise a generic model checking platform based on the method proposed in [36]. If we consider combining the aspects discussed above, a probabilistic timed automaton will be sufficient to construct the combined model. In this case, we only need to introduce special tags to distinguish the transitions of component logics. We can then verify properties expressing multidimensional behaviour using component model checkers of constituent logics. In this way, we do not need to implement a new model checker. We can devise the model checking engine based on individual model checkers, for example, SPIN, UPPAAL, and PRISM.

8. Conclusion

In this paper, we have presented a methodology to make probabilistic model checking more intuitive and accessible for nonexpert users. As part of our methodology, we have proposed the Property Generator tool automatically generating formal properties using natural language statements. As for the model construction, we have used the Drawing Prism tool, providing a GUI that allows us to construct probabilistic state machines from which the model code is automatically generated. In this way, nonexperts can model their systems and express their requirements without any knowledge of the modelling and property specification languages and can analyse their systems without having any expertise.

In our methodology, we have also devised a strategy which reduces the state space of probabilistic models, using the Daikon invariant detector, to increase the performance of model checking and relieve the state explosion problem.

At the moment, the tools discussed and presented in this paper are standalone. As future work, we will integrate these tools to create a software suit. We already have the methods for two features discussed in Section 7. They will also be integrated into the toolset.

Conflict of Interests

The author declares no conflict of interests regarding the publication of this paper.

Acknowledgments

This work is partially supported by the EPSRC (UK) funded ROADBLOCK Project, a joint Project with Dr. M. Gheorghe at the University of Sheffield (EP/I031812/1), Dr. N. Krasnogor at the University of Nottingham (EP/I031642), and Dr. S. Kalvala at the University of Warwick (EP/I03157X).

References

[1] E. Clarke, O. Grumberg, and D. Peled, *Model Checking*, MIT Press, 1999.

[2] A. Cimatti, E. Clarke, F. Giunchiglia, and M. Roveri, " Nusmv : a new symbolic model verifier," in *Proceedings of International Conference on Computer-Aided Verification (CAV '99)*, pp. 495–499, Trento, Italy, 1999.

[3] G. J. Holzmann, *The Spin Model Checker*, Addison-Wesley, Boston, Mass, USA, 2003.

[4] J. Bengtsson, K. G. Larsen, F. Larsson, P. Pettersson, and W. Yi, "UPPAAL—a tool suite for automatic verification of real time systems," in *Proceedings of Workshop on Verification and Control of Hybrid Systems III*, vol. 1066 of *Lecture Notes in Computer Science*, pp. 232–243, Springer, New York, NY, USA, 1995.

[5] M. Bozga, C. Daws, O. Maler, A. Olivero, S. Tripakis, and S. Yovine, "Kronos: a model-checking tool for realtime systems," in *Proceedings of the 10th International Conference on Computer Aided Verification (CAV '98)*, pp. 546–550, Springer, New York, NY, USA, 1998.

[6] S. Konur, M. Fisher, S. Dobson, and S. Knox, "Formal verification of a pervasive messaging system," *Formal Aspects of Computing*, vol. 2013, 18 pages, 2013.

[7] M. Kwiatkowska, G. Norman, and D. Parker, "Quantitative analysis with the pro babilistic model checker PRISM," *Electronic Notes in Theoretical Computer Science*, vol. 153, no. 2, pp. 5–31, 2006.

[8] A. Hinton, M. Kwiatkowska, G. Norman, and D. Parker, "PRISM: a tool for automatic verification of probabilistic systems," in *Tools and Algorithms for the Construction and Analysis of Systems: Proceedings of the 12th International Conference, TACAS 2006, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2006, Vienna, Austria, March 25–April 2, 2006*, vol. 3920 of *Lecture Notes in Computer Science*, pp. 441–444, Springer, Berlin, Germany, 2006.

[9] J. Katoen, I. S. Zapreev, E. M. Hahn, H. Hermanns, and D. N. Jansen, "The ins and outs of the probabilistic model checker MRMC," *Performance Evaluation*, vol. 68, no. 2, pp. 90–104, 2011.

[10] H. L. Younes, "Ymer: a statistical model checker," in *Computer Aided Verification*, vol. 3576 of *Lecture Notes in Computer Science*, pp. 429–433, Springer, Berlin, Germany, 2005.

[11] K. Sen, M. Viswanathan, and G. Agha, "On statistical model checking of stochastic systems," in *Proceedings of the 17th International Conference on Computer Aided Verification (CAV '05)*, vol. 3576 of *Lecture Notes in Computer Science*, pp. 266–280, Springer, 2005.

[12] M. B. Dwyer, G. S. Avrunin, and J. C. Corbett, "Patterns in property specifications for finite-state verification," in *Proceedings of the International Conference on Software Engineering (ICSE '99)*, pp. 411–420, ACM, May 1999.

[13] H. Hansson and B. Jonsson, "A logic for reasoning about time and reliability," *Formal Aspects of Computing*, vol. 6, no. 5, pp. 512–535, 1994.

[14] C. Baier, B. Haverkort, H. Hermanns, and J. Katoen, "Model-checking algorithms for continuous-time Markov chains," *IEEE Transactions on Software Engineering*, vol. 29, no. 6, pp. 524–541, 2003.

[15] PRISM, *Probabilistic Symbolic Model Checker*, 2013, http://www.prismmodelchecker.org/.

[16] M. Kwiatkowska, G. Norman, and D. Parker, "Prism: probabilistic model checking for performance and reliability analysis," *ACM SIGMETRICS Performance Evaluation Review*, vol. 36, no. 4, pp. 40–45, 2009.

[17] MRMC, "Markov Reward Mmodel Checker, Version 1.5," 2011, http://www.mrmc-tool.org/downloads/MRMC/Specs/MRMC_Manual_1.5.pdf.

[18] A. Legay, B. Delahaye, and S. Bensalem, "Statistical model checking: an overview," in *Runtime Verification*, vol. 6418 of *Lecture Notes in Computer Science*, pp. 122–135, Springer, 2010.

[19] F. Zhang, *A swarm-checker, a robot swarm front-end for PRISM [M.S. thesis]*, Department of Computer Science, University of Liverpool, Liverpool, UK, 2010.

[20] A. Ramirez, P. Vanpeperstraete, A. Rueckert et al., *ArgoUML User Manual*, 2010, http://argouml-stats.tigris.org/documentation/manual-0.30/.

[21] M. D. Ernst, J. H. Perkins, P. J. Guo, C. Pacheco, M. S. Tschantz, and C. Xiao, "The Daikon system for dynamic detection of likely invariants," *Science of Computer Programming*, vol. 69, no. 1–3, pp. 35–45, 2007.

[22] F. Bernardini, M. Gheorghe, F. J. Romero-Campero, and N. Walkinshaw, "A hybrid approach to modeling biological systems," in *Membrane Computing*, vol. 4860 of *Lecture Notes in Computer Science*, pp. 138–159, Springer, Berlin, Germany, 2007.

[23] M. Gheorghe, F. Ipate, R. Lefticaru, and C. Dragomir, "An integrated approach to P systems formal verification," in *Membrane Computing*, vol. 6501, pp. 226–239, Springer, Berlin, Germany, 2011.

[24] R. Lefticaru, F. Ipate, L. Valencia-Cabrera et al., "Towards an integrated approach for model simulation, property extraction and verification of P systems," in *Proceedings of the 10th Brainstorming Week on Membrane Computing*, vol. 1, pp. 291–318, Sevilla, Spain, January 2012.

[25] V. Gruhn and R. Laue, "Patterns for timed property specifications," *Electronic Notes in Theoretical Computer Science*, vol. 153, no. 2, pp. 117–133, 2006.

[26] S. Konrad and B. Cheng, "Real-time specification patterns," in *Proceedings of the 27th International Conference on Software Engineering (ICSE '05)*, pp. 372–381, St. Louis, Mo, USA, May 2005.

[27] P. Bellini, P. Nesi, and D. Rogai, "Expressing and organizing real-time specification patterns via temporal logics," *Journal of Systems and Software*, vol. 82, no. 2, pp. 183–196, 2009.

[28] L. Grunske, "Specification patterns for probabilistic quality properties," in *Proceedings of the 30th International Conference on Software Engineering (ICSE '08)*, pp. 31–40, ACM, May 2008.

[29] J. Blakes, J. Twycross, S. Konur, F. Romero-Campero, N. Krasnogor, and M. Gheorghe, "Infobiotics workbench: A p systems based tool for systems and synthetic biology," in *Applications of Membrane Computing in Systems and Synthetic Biology*, vol. 7 of *Emergence, Complexity and Computation*, pp. 1–41, Springer, 2014.

[30] C. Dragomir, F. Ipate, S. Konur, R. Lefticaru, and M. Laurentiu, "Model checking Kernel P systems," in *Proceedings of the 14th International Conference on Membrane Computing (CMC '13)*, pp. 131–152, Chisinau, Moldova, 2013.

[31] W. Liu, A. Winfield, and J. Sa, "Modelling swarm robotic systems: a study in collective foraging," in *Proceedings of the Towards Autonomous Robotic Systems (TAROS '07)*, pp. 25–32, 2007.

[32] S. Konur, C. Dixon, and M. Fisher, "Formal verification of probabilistic swarm behaviours," in *Swarm Intelligence*, vol. 6234 of *Lecture Notes in Computer Science*, pp. 440–447, Springer, Berlin, Germany, 2010.

[33] S. Konur, C. Dixon, and M. Fisher, "Analysing robot swarm behaviour via probabilistic model checking," *Robotics and Autonomous Systems*, vol. 60, no. 2, pp. 199–213, 2012.

[34] R. Alur, C. Courcoubetis, and D. Dill, "Model-checking in dense real-time," *Information and Computation*, vol. 104, no. 1, pp. 2–34, 1993.

[35] M. Kwiatkowska, G. Norman, R. Segala, and J. Sproston, "Automatic verification of real-time systems with discrete probability distributions," *Theoretical Computer Science*, vol. 282, no. 1, pp. 101–150, 2002.

[36] S. Konur, M. Fisher, and S. Schewe, "Combined model checking for temporal, probabilistic, and real-time logics," *Theoretical Computer Science*, vol. 503, pp. 61–88, 2013.

Hindawi Publishing Corporation
Journal of Applied Mathematics
Volume 2014, Article ID 857186, 9 pages
http://dx.doi.org/10.1155/2014/857186

Research Article

Multigranulations Rough Set Method of Attribute Reduction in Information Systems Based on Evidence Theory

Minlun Yan

Department of Mathematics and Applied Mathematics, Lianyungang Teachers College, Lianyungang 222006, China

Correspondence should be addressed to Minlun Yan; haitouyan@163.com

Received 4 February 2014; Revised 16 June 2014; Accepted 17 June 2014; Published 1 July 2014

Academic Editor: Xiaojing Yang

Attribute reduction is one of the most important problems in rough set theory. However, from the granular computing point of view, the classical rough set theory is based on a single granulation. It is necessary to study the issue of attribute reduction based on multigranulations rough set. To acquire brief decision rules from information systems, this paper firstly investigates attribute reductions by combining the multigranulations rough set together with evidence theory. Concepts of belief and plausibility consistent set are proposed, and some important properties are addressed by the view of the optimistic and pessimistic multigranulations rough set. What is more, the multigranulations method of the belief and plausibility reductions is constructed in the paper. It is proved that a set is an optimistic (pessimistic) belief reduction if and only if it is an optimistic (pessimistic) lower approximation reduction, and a set is an optimistic (pessimistic) plausibility reduction if and only if it is an optimistic (pessimistic) upper approximation reduction.

1. Introduction

Rough set theory, originated by Pawlak in the early 1980s [1, 2], is an extension of the classical set theory and can be regarded as a soft computing tool to handle imprecision, vagueness, and uncertainty in date analysis. The theory has been found successful in applications, especially in the field of pattern recognition [3], medical diagnosis [4], data mining [5, 6], conflict analysis [7], algebra [8, 9], and other fields [10, 11]. Recently, the theory has generated a great deal of interest among more and more researchers.

Recently, several extensions of the rough set model have been proposed in terms of various requirements, such as the variable precision rough set (VPRS) model [12], the Bayesian rough set model [13], the fuzzy rough set model, and the rough fuzzy set model [14–16]. Equivalence relation is a basic notion in Pawlak's rough set model. The equivalence classes are employed to construct the lower and upper approximations of an arbitrary subset of the universe of discourse. However, the equivalence relation is a very restrictive condition that may limit applications of rough set. Hence, a variety of extensions of Pawlak's rough set were proposed by employing a more general mathematical concept, for example, arbitrary binary relations [17–19], neighborhood systems and Boolean algebras [20, 21], and partitions and coverings of the universe of discourse [22, 23]. In the view of granular computing (proposed by Zadeh [24]), a general concept described by a set is always characterized via the so-called lower and upper approximations under a single granulation; that is, the concept is depicted by known knowledge induced from a single relation (such as equivalence relation, tolerance relation, and reflexive relation) on the universe.

Since each set of the information granules can be considered as a granulation space, then one can call two or more than two information granules as multigranulations space. To make it more widely to apply the rough set theory in practical applications, Qian and Liang extended Pawlak's single-granulation rough set model to a multigranulations rough set model [25]. Multigranulations rough set was initially proposed by Qian and Liang [25], and later many researchers have extended the multigranulations rough set to the generalized multigranulations rough set. Xu et al. developed a variable multigranulations rough set model [26], a fuzzy multigranulations rough set model [27], a generalized multigranulations rough set approach [28], and a multigranulations rough set model in ordered information systems [29]. Yang et al. proposed the hierarchical structure properties of the multigranulations rough set [30–33] and multigranulations rough set in incomplete information system [34]. Lin et al. presented a neighborhood-based multigranulations rough set [35]. Qian et al. discussed the decision-theoretic rough set theory based on Bayesian decision procedure into the multigranulations [36]. She and He explored the topological structures and the properties of multigranulations rough set [37] and many others [38, 39].

Another important method used to deal with uncertainty in information systems is the Dempster-Shafer theory of evidence. It was originated by Dempster's concept of lower and upper probabilities [40] and extended by Shafer as a theory [41]. The basic representational structure in the theory is a belief structure, which can derive dual pairs of belief and plausibility functions. Subsequently, Zadeh generalized the Dempster-Shafer theory to the fuzzy environment based on his work on the concepts of information granularity [42] and the theory of possibility [43]. So as to evaluate the degrees of belief in fuzzy events, many authors have enriched the Dempster-Shafer theory in different ways. Interested readers can refer to [44] for a summary of some of these generalizations.

There are strong connections between rough set theory and Dempster-Shafer theory of evidence. It has been demonstrated that various belief structures are associated with various rough approximation spaces such that the different dual pairs of lower and upper approximation operators induced by rough approximation spaces may be used to interpret the corresponding dual pairs of belief and plausibility functions induced by belief structures [45–48].

It is well known that attribute reduction is one of the hot research topics of rough set theory. There are many attributes in information system in general. But some attributes are not always needed based on the various reasons. Several kinds of attribute reductions such as upper approximation reductions, lower approximation reductions, and positive region reductions were discussed in a decision system according to different requirements [49–51]. By now much study on this area had been reported and many useful results were obtained [50, 52–55]. While in our real life, we may face some problems in which the existing reductions cannot be disposed, on this situation, some new reductions are needed. In this paper, we attempt to investigate attribute reduction in multigranulations rough set based on evidence theory and its strong relations with existing reductions.

The organization of the rest of this paper is as follows. In Section 2, we give some basic concepts of information systems and Pawlak's rough set, multigranulations rough set, and evidence theory. In Section 3, evidence theory in optimistic multigranulations rough set and pessimistic multigranulations rough set has been constructed, and some important properties are discussed. In Section 4, we introduce the optimistic and pessimistic multigranulations rough set method of belief and plausibility reductions; then we also combine the belief structure with the lower and upper approximations. And in Section 5, we conclude the paper with a summary and outlook for further research.

2. Preliminaries

Throughout this paper, we assume that the universe is a nonempty finite set.

An information system is an order triple $S = (U, \mathrm{AT}, f)$, where $U = \{x_1, x_2, \dots, x_n\}$ is a nonempty finite set of objects, $\mathrm{AT} = \{a_1, a_2, \dots, a_m\}$ is a nonempty finite set of conditional attributes, and for any $a_i \in \mathrm{AT}$, $f_{a_i} : U \to V_{a_i}$ is a map, where V_{a_i} is the domain of the attribute a_i. In particular, a target information system is given by $S = (U, \mathrm{AT}, f, D, g)$, where $D = \{d_1, d_2, \dots, d_p\}$ is a nonempty finite set of decision attributes, and for any $d_j \in D$, $g_{d_j} : U \to V_{d_j}$ is a map, where V_{d_j} is the domain of the attribute d_j. In general, a target information system is consistent, if the partitions induced from the set of condition attributes AT are finer than the partitions induced from the set of decision attributes D. Otherwise, it is inconsistent.

For an information system, any attribute domain V_a may contain special symbol "$*$" to represent that the value of an attribute is unknown. Here, we assume that an object $x \in U$ possesses only one value for an attribute a, $a \in \mathrm{AT}$. Thus, if the value of an attribute a is missing, then the attribute value is the symbol "$*$", and the real value of the attribute must be from the set $V_a \setminus \{*\}$. Any domain value different from "$*$" will be called regular. A system in which values of all attributes for all objects from U are regular (known) is called complete; otherwise it is called incomplete [56, 57].

Throughout this paper, we assume that the information system is the complete information system.

Suppose that $S = (U, \mathrm{AT}, f)$ is an information system, $R_A = \{(x, y) \mid f_{a_i}(x) = f_{a_i}(y), \forall a_i \in A\}$; let U/R_A be a partition of U induced by the attribute subset $A \subseteq \mathrm{AT}$. For any $x \in U$, $[x]_A = \{y \mid (x, y) \in R_A\}$; more information can be found in [58–60].

In the multigranulations rough set model, unlike Pawlak's rough set theory, the target concept is approximated via multiple partitions induced by multiple equivalence relations. Suppose that $S = (U, \mathrm{AT}, f)$ is an information system, $A_1, A_2, \dots, A_s$ $(s \le 2^{|\mathrm{AT}|})$ are attribute subsets, and $X \subseteq U$. Then the optimistic multigranulations lower approximation and upper approximation of X related to $A_1, A_2, \dots, A_s$ in U are defined as follows:

$$\begin{aligned} \underline{\mathrm{OM}_{\sum_{i=1}^{s} A_i}}(X) &= \left\{x \mid \vee\left([x]_{A_i} \subseteq X\right)\right\}, \\ \overline{\mathrm{OM}_{\sum_{i=1}^{s} A_i}}(X) &= \left\{x \mid \wedge\left([x]_{A_i} \cap X \neq \emptyset\right)\right\}. \end{aligned} \tag{1}$$

And the pessimistic multigranulations lower approximation and upper approximation of X related to $A_1, A_2, \dots, A_s$ in U are defined as follows:

$$\begin{aligned} \underline{\mathrm{PM}_{\sum_{i=1}^{s} A_i}}(X) &= \left\{x \mid \wedge\left([x]_{A_i} \subseteq X\right)\right\}, \\ \overline{\mathrm{PM}_{\sum_{i=1}^{s} A_i}}(X) &= \left\{x \mid \vee\left([x]_{A_i} \cap X \neq \emptyset\right)\right\}. \end{aligned} \tag{2}$$

More detailed introductions can be seen in [58, 59, 61, 62].

In evidence theory [40, 41], a mass function of a universe U can be defined by a map $m : P(U) \to [0,1]$, and this mass function satisfies two axioms:

$$\begin{aligned} &(M1) \quad m(\emptyset) = 0, \\ &(M2) \quad \sum_{X \subseteq U} m(X) = 1. \end{aligned} \tag{3}$$

The value $m(X)$ represents the degree of belief that a specific element of U belongs to set X but not to any particular subset of X.

If $m(X) > 0$, then X is called a focal element. The family of all focal elements of m are denoted by M. Then the pair (M, m) is called a belief structure.

In information systems, each belief structure can derive a pair of belief and plausibility functions based on classical equivalence relation.

Definition 1 (see [40, 41]). Let (M, m) be a belief structure. A set function $\mathrm{Bel} : P(U) \to [0,1]$ is referred to as a belief function on U, if for any $X \in P(U)$,

$$\mathrm{Bel}(X) = \sum_{Y \subseteq X} m(Y). \tag{4}$$

A set function $\mathrm{Pl} : P(U) \to [0,1]$ is referred to as a plausibility function on U, if for any $X \in P(U)$,

$$\mathrm{Pl}(X) = \sum_{Y \cap X \neq \emptyset} m(Y). \tag{5}$$

Remark 2. The above definition about belief and plausibility functions can also be defined as follows.

A set function $\mathrm{Bel} : P(U) \to [0,1]$ is referred to as a belief function on U, if it satisfies the following three axioms [48]:

(1) $\mathrm{Bel}(\emptyset) = 0$,

(2) $\mathrm{Bel}(U) = 1$,

(3) for any positive integer n and every collection $X_1, X_2, \ldots, X_n \subseteq U$,

$$\mathrm{Bel}\left(\bigcup_{i=1}^{n} X_i\right) \geq \sum_{\emptyset \neq J \subseteq \{1,2,\ldots,n\}} (-1)^{|J|+1} \mathrm{Bel}\left(\bigcap_{i \in J} X_i\right). \tag{6}$$

A set function $\mathrm{Pl} : P(U) \to [0,1]$ is referred to as a plausibility function on U, if it satisfies the following three axioms [48]:

(1) $\mathrm{Pl}(\emptyset) = 0$,

(2) $\mathrm{Pl}(U) = 1$,

(3) for any positive integer n and every collection $X_1, X_2, \ldots, X_n \subseteq U$,

$$\mathrm{Pl}\left(\bigcap_{i=1}^{n} X_i\right) \leq \sum_{\emptyset \neq J \subseteq \{1,2,\ldots,n\}} (-1)^{|J|+1} \mathrm{Pl}\left(\bigcup_{i \in J} X_i\right). \tag{7}$$

From the definition and remark above, one can test and verify belief and plausibility functions by validating the three axioms above, respectively.

3. Evidence Theory in Multigranulations Rough Set

In this section, we will introduce the evidence theory in the optimistic multigranulations rough set and pessimistic multigranulations rough set and discuss some important properties of evidence theory in information system.

Definition 3. Let $S = (U, \mathrm{AT}, f)$ be an information system, $A_1, A_2, \ldots, A_s \subseteq \mathrm{AT}$ $(s \leq 2^{|\mathrm{AT}|})$, and let R_i be the equivalence relation associated with A_i. For any $X \in U/R_i$, a mass function m_i of S can be defined as $m_i(X) = |X|/|U|$, where $|X|$ denotes the cardinality of a set X.

By the above definition, we can easily find that a mass function of information system satisfies two basic axioms. That is to say, for any $X \in U/R_i$ in information system, the following two axioms hold directly:

$$\begin{aligned} &(M1) \quad m_i(\emptyset) = 0, \\ &(M2) \quad \sum_{X \subseteq U/R_i} m_i(X) = 1. \end{aligned} \tag{8}$$

Similarly, the family of all focal elements of m is denoted by M in optimistic multigranulations rough set. The pair (M, m) is called a belief structure of the optimistic multigranulations rough set in information system; a pair of belief and plausibility function in the optimistic multigranulations rough set can be derived immediately.

Definition 4. Let $S = (U, \mathrm{AT}, f)$ be an information system and (M, m) a belief structure.

(1) A set function $\mathrm{Bel} : P(U) \to [0,1]$ is referred to as an optimistic belief function on U, if for any $X \in P(U)$,

$$\mathrm{Bel}^{o}_{\sum_{i=1}^{s} A_i}(X) = \sum_{Y \subseteq X, Y \in U/R_{A_i} (\exists A_i \subseteq \mathrm{AT})} m_i(Y). \tag{9}$$

A set function $\mathrm{Pl} : P(U) \to [0,1]$ is referred to as an optimistic plausibility function on U, if for any $X \in P(U)$,

$$\mathrm{Pl}^{o}_{\sum_{i=1}^{s} A_i}(X) = \sum_{Y \cap X \neq \emptyset, Y \in U/R_{A_i} (\forall A_i \subseteq \mathrm{AT})} m_i(Y). \tag{10}$$

(2) A set function $\mathrm{Bel} : P(U) \to [0,1]$ is referred to as a pessimistic belief function on U, if for any $X \in P(U)$,

$$\mathrm{Bel}^{P}_{\sum_{i=1}^{s} A_i}(X) = \sum_{Y \subseteq X, Y \in U/R_{A_i} (\forall A_i \subseteq \mathrm{AT})} m_i(Y). \tag{11}$$

A set function $\mathrm{Pl} : P(U) \to [0,1]$ is referred to as a pessimistic plausibility function on U, if for any $X \in P(U)$,

$$\mathrm{Pl}^{P}_{\sum_{i=1}^{s} A_i}(X) = \sum_{Y \cap X \neq \emptyset, Y \in U/R_{A_i} (\exists A_i \subseteq \mathrm{AT})} m_i(Y). \tag{12}$$

Theorem 5. *Let $S = (U, AT, f)$ be an information system, for any $X \subseteq U$, $A_1, A_2, \ldots, A_s \subseteq AT$ ($s \leq 2^{|AT|}$), denoted by*

$$\begin{aligned} Bel^{o}_{\sum_{i=1}^{s} A_i}(X) &= \frac{\left|\underline{OM_{\sum_{i=1}^{s} A_i}}(X)\right|}{|U|}, \\ Pl^{o}_{\sum_{i=1}^{s} A_i}(X) &= \frac{\left|\overline{OM_{\sum_{i=1}^{s} A_i}}(X)\right|}{|U|}, \\ Bel^{P}_{\sum_{i=1}^{s} A_i}(X) &= \frac{\left|\underline{PM_{\sum_{i=1}^{s} A_i}}(X)\right|}{|U|}, \\ Pl^{P}_{\sum_{i=1}^{s} A_i}(X) &= \frac{\left|\overline{PM_{\sum_{i=1}^{s} A_i}}(X)\right|}{|U|}. \end{aligned} \tag{13}$$

Then

$Bel^{o}_{\sum_{i=1}^{s} A_i}(X)$ is the optimistic belief function and $Pl^{o}_{\sum_{i=1}^{s} A_i}(X)$ is the optimistic plausibility function of U;

$Bel^{P}_{\sum_{i=1}^{s} A_i}(X)$ is the pessimistic belief function and $Pl^{P}_{\sum_{i=1}^{s} A_i}(X)$ is the pessimistic plausibility function of U.

Proof. We only prove $\mathrm{Bel}^{o}_{\sum_{i=1}^{s} A_i}(X)$ is the optimistic belief function of U; then analogously we can prove $\mathrm{Pl}^{o}_{\sum_{i=1}^{s} A_i}(X)$ is the optimistic plausibility function of U. Consider

(1) $\mathrm{Bel}^{o}_{\sum_{i=1}^{s} A_i}(\emptyset) = |\underline{\mathrm{OM}_{\sum_{i=1}^{s} A_i}}(\emptyset)|/|U| = 0/|U| = 0$,

(2) $\mathrm{Bel}^{o}_{\sum_{i=1}^{s} A_i}(U) = |\underline{\mathrm{OM}_{\sum_{i=1}^{s} A_i}}(U)|/|U| = |U|/|U| = 1$,

(3) for any positive integer n and every collection $X_1, X_2, \ldots, X_n \subseteq U$, we have

$$\begin{aligned} &\mathrm{Bel}^{o}_{\sum_{i=1}^{s} A_i}(X_1 \cup X_2 \cup \cdots \cup X_n) \\ &= \frac{\left|\underline{\mathrm{OM}_{\sum_{i=1}^{s} A_i}}(X_1 \cup X_2 \cup \cdots \cup X_n)\right|}{|U|} \\ &\geq \frac{\left|\underline{\mathrm{OM}_{\sum_{i=1}^{s} A_i}}(X_1) \cup \underline{\mathrm{OM}_{\sum_{i=1}^{s} A_i}}(X_2) \cup \cdots \cup \underline{\mathrm{OM}_{\sum_{i=1}^{s} A_i}}(X_n)\right|}{|U|} \\ &= \sum_{j=1}^{n} \frac{\left|\underline{\mathrm{OM}_{\sum_{i=1}^{s} A_i}}(X_j)\right|}{|U|} \\ &\quad - \sum_{k<j} \frac{\left|\underline{\mathrm{OM}_{\sum_{i=1}^{s} A_i}}(X_k) \cap \underline{\mathrm{OM}_{\sum_{i=1}^{s} A_i}}(X_j)\right|}{|U|} + \cdots + (-1)^{n+1} \\ &\quad \times \frac{\left|\underline{\mathrm{OM}_{\sum_{i=1}^{s} A_i}}(X_1) \cap \underline{\mathrm{OM}_{\sum_{i=1}^{s} A_i}}(X_2) \cap \cdots \cap \underline{\mathrm{OM}_{\sum_{i=1}^{s} A_i}}(X_n)\right|}{|U|} \\ &\geq \sum_{j=1}^{n} \frac{\left|\underline{\mathrm{OM}_{\sum_{i=1}^{s} A_i}}(X_j)\right|}{|U|} - \sum_{k<j} \frac{\left|\underline{\mathrm{OM}_{\sum_{i=1}^{s} A_i}}(X_k \cap X_j)\right|}{|U|} \\ &\quad + \cdots + (-1)^{n+1} \frac{\left|\underline{\mathrm{OM}_{\sum_{i=1}^{s} A_i}}(X_1 \cap X_2 \cdots \cap X_n)\right|}{|U|} \\ &= \sum_{j=1}^{n} \mathrm{Bel}^{o}_{\sum_{i=1}^{s} A_i}(X_j) - \sum_{k<j} \mathrm{Bel}^{o}_{\sum_{i=1}^{s} A_i}(X_k \cap X_j) \\ &\quad + \cdots + (-1)^{n+1} \mathrm{Bel}^{o}_{\sum_{i=1}^{s} A_i}(X_1 \cap X_2 \cap \cdots \cap X_n). \end{aligned} \tag{14}$$

Similarly we can prove that $\mathrm{Bel}^{P}_{\sum_{i=1}^{s} A_i}(X)$ is the pessimistic belief function and $\mathrm{Pl}^{P}_{\sum_{i=1}^{s} A_i}(X)$ is the pessimistic plausibility function of U.

Thus, the theorem is proved. □

From the theorem above, one can get the following properties of the belief function and plausibility function in the optimistic and pessimistic multigranulations rough set.

Theorem 6. *Belief function and plausibility function based on the same belief structure are connected by the dual property in the optimistic multigranulations rough set and the pessimistic multigranulations rough set, respectively,*

(1) *$Bel^{o}_{\sum_{i=1}^{s} A_i}(X) = 1 - Pl^{o}_{\sum_{i=1}^{s} A_i}(\sim X)$,*

(2) *$Bel^{P}_{\sum_{i=1}^{s} A_i}(X) = 1 - Pl^{P}_{\sum_{i=1}^{s} A_i}(\sim X)$.*

Proof. (1) It is clear that $\underline{\mathrm{OM}_{\sum_{i=1}^{s} A_i}}(X) = \sim \overline{\mathrm{OM}_{\sum_{i=1}^{s} A_i}}(\sim X)$, so $|\underline{\mathrm{OM}_{\sum_{i=1}^{s} A_i}}(X)| = |U| - |\overline{\mathrm{OM}_{\sum_{i=1}^{s} A_i}}(\sim X)|$. Then

$$\begin{aligned} \frac{\left|\underline{\mathrm{OM}_{\sum_{i=1}^{s} A_i}}(X)\right|}{|U|} &= \frac{|U| - \left|\overline{\mathrm{OM}_{\sum_{i=1}^{s} A_i}}(\sim X)\right|}{|U|} \\ &= 1 - \frac{\left|\overline{\mathrm{OM}_{\sum_{i=1}^{s} A_i}}(\sim X)\right|}{|U|}. \end{aligned} \tag{15}$$

That is to say

$$\mathrm{Bel}^{o}_{\sum_{i=1}^{s} A_i}(X) = 1 - \mathrm{Pl}^{o}_{\sum_{i=1}^{s} A_i}(\sim X). \tag{16}$$

(2) One can prove $\mathrm{Bel}^{P}_{\sum_{i=1}^{s} A_i}(X) = 1 - \mathrm{Pl}^{P}_{\sum_{i=1}^{s} A_i}(\sim X)$ to be similar. □

Corollary 7. *Let $S = (U, AT, f)$ be an information system, for any $X \subseteq U$, $A_1, A_2, \ldots, A_s \subseteq AT$ ($s \leq 2^{|AT|}$); then*

(1) *$Bel^{o}_{A_i}(X) \leq Bel^{o}_{\sum_{1\leq j\leq s}^{i\neq j} A_i}(X) \leq Bel^{o}_{\sum_{i=1}^{s} A_i}(X) \leq |X|/|U| \leq Pl^{o}_{\sum_{i=1}^{s} A_i}(X) \leq Pl^{o}_{\sum_{1\leq j\leq s}^{i\neq j} A_i}(X) \leq Pl^{o}_{A_i}(X)$,*

(2) *$Bel^{P}_{\sum_{i=1}^{s} A_i}(X) \leq Bel^{P}_{\sum_{1\leq j\leq s}^{i\neq j} A_i}(X) \leq Bel^{P}_{A_i}(X) \leq |X|/|U| \leq Pl^{P}_{A_i}(X) \leq Pl^{P}_{\sum_{1\leq j\leq s}^{i\neq j} A_i}(X) \leq Pl^{P}_{\sum_{i=1}^{s} A_i}(X)$.*

Proof. It can be obtained directly by combining the properties of the optimistic multigranulations rough set, pessimistic multigranulations rough set, and Theorem 5. □

The difference between (1) and (2) in Corollary 7 is mainly because of the difference between the definition of the optimistic and pessimistic multigranulations rough set lower and upper approximations.

Example 8. Table 1 depicts a target information system containing some information about an emporium investment project. *Locus*, *Investment*, and *Population density* are the conditional attributes of the system, and *Decision* is the decision attribute. (In the sequel, L, I, P, and D will stand for *Locus*, *Investment*, *Population density*, and *Decision*, resp.) The attribute domains are as follows: $V_L = \{\text{Good}, \text{Common}, \text{Bad}\}$, $V_I = \{\text{High}, \text{Low}\}$, $V_P = \{\text{Big}, \text{Small}, \text{Medium}\}$, and $V_D = \{\text{Yes}, \text{No}\}$.

Let $X = \{x_1, x_2, x_6, x_8\}$; we have gotten

$$\begin{gathered}
\underline{\mathrm{OM}_L}(X) = \underline{R_L}(X) = \{x_8\}, \\
\underline{\mathrm{PM}_L}(X) = \underline{R_L}(X) = \{x_8\}, \\
\overline{\mathrm{OM}_L}(X) = \overline{R_L}(X) = \{x_1, x_2, \dots, x_8\}, \\
\overline{\mathrm{PM}_L}(X) = \overline{R_L}(X) = \{x_1, x_2, \dots, x_8\}, \\
\underline{\mathrm{OM}_{L+P}}(X) = \{x_1, x_2, x_8\}, \qquad \underline{\mathrm{PM}_{L+P}}(X) = \emptyset, \\
\overline{\mathrm{OM}_{L+P}}(X) = \{x_1, x_2, x_6, x_7, x_8\}, \\
\overline{\mathrm{PM}_{L+P}}(X) = \{x_1, x_2, \dots, x_8\}, \\
\underline{\mathrm{OM}_{L+I+P}}(X) = \{x_1, x_2, x_8\}, \qquad \underline{\mathrm{PM}_{L+I+P}}(X) = \emptyset, \\
\overline{\mathrm{OM}_{L+I+P}}(X) = \{x_1, x_2, x_6, x_7, x_8\}, \\
\overline{\mathrm{PM}_{L+I+P}}(X) = \{x_1, x_2, \dots, x_8\}.
\end{gathered} \tag{17}$$

So, we can calculate

$$\begin{gathered}
\mathrm{Bel}_L^o(X) = \frac{\left|\underline{\mathrm{OM}_L}(X)\right|}{|U|} = \frac{1}{8}, \\
\mathrm{Bel}_L^P(X) = \frac{\left|\underline{\mathrm{PM}_L}(X)\right|}{|U|} = \frac{1}{8}, \\
\mathrm{Pl}_L^o(X) = \frac{\left|\overline{\mathrm{OM}_L}(X)\right|}{|U|} = 1, \\
\mathrm{Pl}_L^P(X) = \frac{\left|\overline{\mathrm{PM}_L}(X)\right|}{|U|} = 1, \\
\mathrm{Bel}_{L+P}^o(X) = \frac{\left|\underline{\mathrm{OM}_{L+P}}(X)\right|}{|U|} = \frac{3}{8}, \\
\mathrm{Bel}_{L+P}^P(X) = \frac{\left|\underline{\mathrm{PM}_{L+P}}(X)\right|}{|U|} = 0, \\
\mathrm{Pl}_{L+P}^o(X) = \frac{\left|\overline{\mathrm{OM}_{L+P}}(X)\right|}{|U|} = \frac{5}{8}, \\
\mathrm{Pl}_{L+P}^P(X) = \frac{\left|\overline{\mathrm{PM}_{L+P}}(X)\right|}{|U|} = 1, \\
\mathrm{Bel}_{L+I+P}^o(X) = \frac{\left|\underline{\mathrm{OM}_{L+I+P}}(X)\right|}{|U|} = \frac{3}{8}, \\
\mathrm{Bel}_{L+I+P}^P(X) = \frac{\left|\underline{\mathrm{PM}_{L+I+P}}(X)\right|}{|U|} = 0, \\
\mathrm{Pl}_{L+I+P}^o(X) = \frac{\left|\overline{OM_{L+I+P}}(X)\right|}{|U|} = \frac{5}{8}, \\
\mathrm{Pl}_{L+I+P}^P(X) = \frac{\left|\overline{\mathrm{PM}_{L+I+P}}(X)\right|}{|U|} = 1.
\end{gathered} \tag{18}$$

Hence, the following is obvious:

$$\begin{aligned}
&\mathrm{Bel}_L^o(X) \le \mathrm{Bel}_{L+P}^o(X) \le \mathrm{Bel}_{L+I+P}^o(X) \\
&\quad \le \frac{|X|}{|U|} \le \mathrm{Pl}_{L+I+P}^o(X) \le \mathrm{Pl}_{L+P}^o(X) \le \mathrm{Pl}_L^o(X), \\
&\mathrm{Bel}_{L+I+P}^P(X) \le \mathrm{Bel}_{L+P}^P(X) \le \mathrm{Bel}_L^P(X) \\
&\quad \le \frac{|X|}{|U|} \le \mathrm{Pl}_L^P(X) \le \mathrm{Pl}_{L+P}^P(X) \le \mathrm{Pl}_{L+I+P}^P(X).
\end{aligned} \tag{19}$$

4. Attribute Reduction Based on Evidence Theory

In this section, we consider the optimistic multiple granulation rough set and the pessimistic multigranulations rough set method of the attribute reductions by introducing the concepts of belief and plausibility reductions in information system and compare them with the existing reductions.

Let $S = (U, \mathrm{AT}, f, D, g)$ be a decision information system, $A \subseteq \mathrm{AT}$, and let R_A and R_D be the equivalence relations of U, which are induced by the conditional attribute set A and the decision attribute set $D = \{d\}$, respectively; we denote

$$\begin{gathered}
\frac{U}{R_A} = \{[x_i]_A \mid x_i \in U\}, \\
\frac{U}{R_D} = \{D_1, D_2, \dots, D_r\}, \\
\overline{\eta}_A^o = \Big(\overline{\mathrm{OM}_{\sum_{A_i \in A} A_i}}(D_1), \overline{\mathrm{OM}_{\sum_{A_i \in A} A_i}}(D_2), \\
\dots, \overline{\mathrm{OM}_{\sum_{A_i \in A} A_i}}(D_r)\Big),
\end{gathered}$$

TABLE 1: A target information system about emporium investment project.

Project	Locus	Investment	Population density	Decision
x_1	Common	High	Big	Yes
x_2	Bad	High	Big	Yes
x_3	Bad	Low	Small	No
x_4	Bad	Low	Small	No
x_5	Bad	Low	Small	No
x_6	Bad	High	Medium	Yes
x_7	Common	High	Medium	No
x_8	Good	High	Medium	Yes

$$\underline{\eta}_A^o = \left(\underline{\mathrm{OM}_{\sum_{A_i\in A} A_i}}(D_1), \underline{\mathrm{OM}_{\sum_{A_i\in A} A_i}}(D_2), \ldots, \underline{\mathrm{OM}_{\sum_{A_i\in A} A_i}}(D_r)\right),$$
$$\overline{\eta}_A^P = \left(\overline{\mathrm{PM}_{\sum_{A_i\in A} A_i}}(D_1), \overline{\mathrm{PM}_{\sum_{A_i\in A} A_i}}(D_2), \ldots, \overline{\mathrm{PM}_{\sum_{A_i\in A} A_i}}(D_r)\right),$$
$$\underline{\eta}_A^P = \left(\underline{\mathrm{PM}_{\sum_{A_i\in A} A_i}}(D_1), \underline{\mathrm{PM}_{\sum_{A_i\in A} A_i}}(D_2), \ldots, \underline{\mathrm{PM}_{\sum_{A_i\in A} A_i}}(D_r)\right), \tag{20}$$

where $[x_i]_A = \{y \in U \mid (x_i, y) \in R_A\}$.

Definition 9 (see [58]). Let $S = (U, \mathrm{AT}, f, D, g)$ be a decision information system in which $A \subseteq \mathrm{AT}$.

(1) If $\overline{\eta}_A^o = \overline{\eta}_{\mathrm{AT}}^o$, then A is referred to as an optimistic upper approximation consistent set of S with respect to the equivalence relation R_A; if A is an optimistic upper approximation consistent set of S and for any $A' \subseteq A$, A' is not the optimistic upper approximation consistent set of S, then A is referred to as an optimistic upper approximation reduction of S with respect to the equivalence relation R_A.

(2) If $\underline{\eta}_A^o = \underline{\eta}_{\mathrm{AT}}^o$, then A is referred to as an optimistic lower approximation consistent set of S with respect to the equivalence relation R_A; if A is an optimistic lower approximation consistent set of S and for any $A' \subseteq A$, A' is not the optimistic lower approximation consistent set of S, then A is referred to as an optimistic lower approximation reduction of S with respect to the equivalence relation R_A.

(3) If $\overline{\eta}_A^P = \overline{\eta}_{\mathrm{AT}}^P$, then A is referred to as a pessimistic upper approximation consistent set of S with respect to the equivalence relation R_A; if A is a pessimistic upper approximation consistent set of S and for any $A' \subseteq A$, A' is not the pessimistic upper approximation consistent set of S, then A is referred to as a pessimistic upper approximation reduction of S with respect to the equivalence relation R_A.

(4) If $\underline{\eta}_A^P = \underline{\eta}_{\mathrm{AT}}^P$, then A is referred to as a pessimistic lower approximation consistent set of S with respect to the equivalence relation R_A; if A is a pessimistic lower approximation consistent set of S and for any $A' \subseteq A$, A' is not the pessimistic lower approximation consistent set of S, then A is referred to as a pessimistic lower approximation reduction of S with respect to the equivalence relation R_A. From the definitions above, one can get the following theorem directly.

Theorem 10. *Let $S = (U, AT, f, D, g)$ be a decision information system in which $A \subseteq AT$; then*

(1) *A is an optimistic upper approximation consistent set of S if and only if, for any $D_j \in U/R_D$, $\overline{OM_{\sum_{A_i\in A} A_i}}(D_j) = \overline{R_{AT}}(D_j)$ holds;*

(2) *A is an optimistic lower approximation consistent set of S if and only if, for any $D_j \in U/R_D$, $\underline{OM_{\sum_{A_i\in A} A_i}}(D_j) = \underline{R_{AT}}(D_j)$ holds;*

(3) *A is a pessimistic upper approximation consistent set of S if and only if, for any $D_j \in U/R_D$, $\overline{PM_{\sum_{A_i\in A} A_i}}(D_j) = \overline{R_{AT}}(D_j)$ holds;*

(4) *A is a pessimistic lower approximation consistent set of S if and only if, for any $D_j \in U/R_D$, $\underline{PM_{\sum_{A_i\in A} A_i}}(D_j) = \underline{R_{AT}}(D_j)$ holds.*

Proof. It can be derived easily from the definition of the optimistic upper and lower approximation reduction and the pessimistic upper and lower approximation reduction. □

Definition 11. Let $S = (U, \mathrm{AT}, f, D, g)$ be a decision information system in which $A \subseteq \mathrm{AT}$.

(1) If for any $X \in U/R_{\mathrm{AT}}$, $\mathrm{Bel}^o_{\sum_{A_i\in A} A_i}(X) = \mathrm{Bel}^o_{\mathrm{AT}}(X)$, then A is referred to as an optimistic belief consistent set of S; if A is an optimistic belief consistent set of S and for any $A' \subseteq A$, A' is not the optimistic belief consistent set of S, then A is referred to as an optimistic belief reduction of S.

(2) If for any $X \in U/R_{\mathrm{AT}}$, $\mathrm{Pl}^o_{\sum_{A_i\in A} A_i}(X) = \mathrm{Pl}^o_{\mathrm{AT}}(X)$, then A is referred to as an optimistic plausibility consistent set of S; if A is an optimistic plausibility consistent

set of S and for any $A' \subseteq A$, A' is not the optimistic plausibility consistent set of S, then A is referred to as an optimistic plausibility reduction of S.

(3) If for any $X \in U/R_{\mathrm{AT}}$, $\mathrm{Bel}^{P}_{\sum_{A_i\in A} A_i}(X) = \mathrm{Bel}^{P}_{\mathrm{AT}}(X)$, then A is referred to as a pessimistic belief consistent set of S; if A is a pessimistic belief consistent set of S and for any $A' \subseteq A$, A' is not the pessimistic belief consistent set of S, then A is referred to as a pessimistic belief reduction of S.

(4) If for any $X \in U/R_{\mathrm{AT}}$, $\mathrm{Pl}^{P}_{\sum_{A_i\in A} A_i}(X) = \mathrm{Pl}^{P}_{\mathrm{AT}}(X)$, then A is referred to as a pessimistic plausibility consistent set of S; if A is a pessimistic plausibility consistent set of S and for any $A' \subseteq A$, A' is not the pessimistic plausibility consistent set of S, then A is referred to as a pessimistic plausibility reduction of S.

Theorem 12. *Let $S = (U, AT, f, D, g)$ be a decision information system in which $A \subseteq AT$; then*

(1) *A is an optimistic belief consistent set of S if and only if A is an optimistic lower approximation consistent set of S;*

(2) *A is an optimistic plausibility consistent set of S if and only if A is an optimistic upper approximation consistent set of S;*

(3) *A is a pessimistic belief consistent set of S if and only if A is a pessimistic lower approximation consistent set of S;*

(4) *A is a pessimistic plausibility consistent set of S if and only if A is a pessimistic upper approximation consistent set of S.*

Proof. (1) Assume that A is an optimistic belief consistent set of S; for any $X \in U/R_{\mathrm{AT}}$, we have $\mathrm{Bel}^{o}_{\sum_{A_i\in A} A_i}(X) = \mathrm{Bel}^{o}_{\mathrm{AT}}(X)$. That is to say, $|\underline{\mathrm{OM}_{\sum_{A_i\in A} A_i}}(X)| = |\underline{R_{\mathrm{AT}}}(X)|$. Then by the definition of the optimistic lower approximation we have $[x]_{\mathrm{AT}} \subseteq [x]_{A_i}$; then $\underline{\mathrm{OM}_{\sum_{A_i\in A} A_i}}(X) \subseteq \underline{R_{\mathrm{AT}}}(X)$. So $\underline{\mathrm{OM}_{\sum_{A_i\in A} A_i}}(X) = \underline{R_{\mathrm{AT}}}(X)$. By Theorem 10, we obtain A is an optimistic lower approximation consistent set of S.

Conversely, if A is an optimistic lower approximation consistent set of S, for any $D_j \in U/R_D$, we have $\underline{\mathrm{OM}_{\sum_{A_i\in A} A_i}}(D_j) = \underline{R_{\mathrm{AT}}}(D_j)$. Then $|\underline{\mathrm{OM}_{\sum_{A_i\in A} A_i}}(D_j)|/|U| = |\underline{R_{\mathrm{AT}}}(D_j)|/|U|$. That is to say, $\mathrm{Bel}^{o}_{\sum_{A_i\in A} A_i}(D_j) = \mathrm{Bel}^{o}_{\mathrm{AT}}(D_j)$. Thus A is an optimistic belief consistent set of S.

(2) Assume that A is an optimistic plausibility consistent set of S; for any $X \in U/R_{\mathrm{AT}}$, we have $\mathrm{Pl}^{o}_{\sum_{A_i\in A} A_i}(X) = \mathrm{Pl}^{o}_{\mathrm{AT}}(X)$. That is to say, $|\overline{\mathrm{OM}_{\sum_{A_i\in A} A_i}}(X)| = |\overline{R_{\mathrm{AT}}}(X)|$. By the definition of the optimistic upper approximation we have, $[x]_{\mathrm{AT}} \subseteq [x]_{A_i}$. Then $\overline{\mathrm{OM}_{\sum_{A_i\in A} A_i}}(X) \supseteq \overline{R_{\mathrm{AT}}}(X)$. So $\overline{\mathrm{OM}_{\sum_{A_i\in A} A_i}}(X) = \overline{R_{\mathrm{AT}}}(X)$. By Theorem 10, we obtain A is an optimistic upper approximation consistent set of S.

Conversely, if A is an optimistic upper approximation consistent set of S, for any $D_j \in U/R_D$, we have $\overline{\mathrm{OM}_{\sum_{A_i\in A} A_i}}(D_j) = \overline{R_{\mathrm{AT}}}(D_j)$. Then $|\overline{\mathrm{OM}_{\sum_{A_i\in A} A_i}}(D_j)|/|U| = |\overline{R_{\mathrm{AT}}}(D_j)|/|U|$. That is to say, $\mathrm{Pl}^{o}_{\sum_{A_i\in A} A_i}(D_j) = \mathrm{Pl}^{o}_{\mathrm{AT}}(D_j)$. Thus A is an optimistic plausibility consistent set of S.

(3) It is straightforward by (1).

(4) It is straightforward by (2).

Hence, the proof is completed. □

Table 2: A system about emporium investment project.

Project	Decision
x_1	Yes
x_2	Yes
x_3	No
x_4	No
x_5	No
x_6	Yes
x_7	No
x_8	Yes

Corollary 13. *Let $S = (U, AT, f, D, g)$ be a decision information system in which $A \subseteq AT$; then*

(1) *A is an optimistic belief reduction of S if and only if A is an optimistic lower approximation reduction of S;*

(2) *A is an optimistic plausibility reduction of S if and only if A is an optimistic upper approximation reduction of S;*

(3) *A is a pessimistic belief reduction of S if and only if A is a pessimistic lower approximation reduction of S;*

(4) *A is a pessimistic plausibility reduction of S if and only if A is a pessimistic upper approximation reduction of S.*

We have shown that the belief and plausibility reduction are the same with lower and upper approximation reduction. One may deal with some issue in which the lower and upper approximation reduction cannot be explained, while the belief and plausibility reduction can be well explained. Just like this, indefinite integral and definite integral are two basic problems in the integral calculus. Indefinite integral is the inverse operation of differentiation, while definite integral is the limit of a particular type of the sum. There are both difference and connection between them. The definite integral and the indefinite integral can be connected in theory using Newton-leibniz formula, while some functions still can be integrated, though they have no antiderivative on closed interval. Under this circumstance, indefinite integral cannot be evaluated by Newton-leibniz formula. Next we use an example to illustrate this problem.

Example 14. Table 2 depicts a system containing some information about an emporium investment project. All projects are $U = \{x_1, x_2, \ldots, x_8\}$, and *Decision* is the decision attribute. (In the sequel, D will stand for *Decision*.)

In this example, there are no conditional attributes. So we cannot use the lower and upper approximations to deal with this system. Based on evidence theory, each object can be given corresponding confidence degree according to the existing information. And this confidence degree after normalization can be used as a new mass function. Then we can apply the evidence theory to dispose this system. Until now, we have not found a good method to evaluate the confidence degree. This will be an important issue of our research in the future.

5. Conclusions

It is well known that rough set theory has been regarded as a generalization of the classical set theory in some cases. In this paper, we have combined the rough set theory and evidence theory, in order to study the problem of attribute reductions. By introducing mass function based on the multigranulations rough set, we have considered the notions of the multigranulations rough set method of belief and plausibility reductions. What is more, we have found that a set is an optimistic (pessimistic) belief reduction if and only if it is an optimistic (pessimistic) lower approximation reduction, and a set is an optimistic (pessimistic) plausibility reduction if and only if it is an optimistic (pessimistic) upper approximation reduction by optimistic and pessimistic frames, respectively. And we will investigate the specific application of theories obtained in our further study.

Conflict of Interests

The author declares that there is no conflict of interests regarding the publication of this paper.

References

[1] Z. Pawlak, "Rough sets," *International Journal of Computer and Information Sciences*, vol. 11, no. 5, pp. 341–356, 1982.

[2] Z. Pawlak, *Rough Set: Theoretical Aspects of Reasoning about Data*, Kluwer Academic Publishers, Dordrecht, The Netherlands, 1991.

[3] R. W. Swiniarski and A. Skowron, "Rough set methods in feature selection and recognition," *Pattern Recognition Letters*, vol. 24, no. 6, pp. 833–849, 2003.

[4] S. Tsumoto, "Automated extraction of medical expert system rules from clinical databases based on rough set theory," *Information Sciences*, vol. 112, no. 1–4, pp. 67–84, 1998.

[5] V. S. Ananthanarayana, M. Narasimha Murty, and D. K. Subramanian, "Tree structure for efficient data mining using rough sets," *Pattern Recognition Letters*, vol. 24, no. 6, pp. 851–862, 2003.

[6] C. Chan, "A rough set approach to attribute generalization in data mining," *Information Sciences*, vol. 107, no. 1–4, pp. 169–176, 1998.

[7] Z. Pawlak, "Some remarks on conflict analysis," *European Journal of Operational Research*, vol. 166, no. 3, pp. 649–654, 2005.

[8] W. Cheng, Z. W. Mo, and J. Wang, "Notes on "the lower and upper approximations in a fuzzy group" and 'rough ideals in semigroups,'" *Information Sciences*, vol. 177, no. 22, pp. 5134–5140, 2007.

[9] B. Davvaz and M. Mahdavipour, "Roughness in modules," *Information Sciences*, vol. 176, no. 24, pp. 3658–3674, 2006.

[10] I. Düntsch and G. Gediga, "Uncertainty measures of rough set prediction," *Artificial Intelligence*, vol. 106, no. 1, pp. 109–137, 1998.

[11] G. Jeon, D. Kim, and J. Jeong, "Rough sets attributes reduction based expert system in interlaced video sequences," *IEEE Transactions on Consumer Electronics*, vol. 52, no. 4, pp. 1348–1355, 2006.

[12] W. Ziarko, "Variable precision rough set model," *Journal of Computer and System Sciences*, vol. 46, no. 1, pp. 39–59, 1993.

[13] D. Ślęzak and W. Ziarko, "The investigation of the Bayesian rough set model," *International Journal of Approximate Reasoning*, vol. 40, no. 1-2, pp. 81–91, 2005.

[14] D. Dubois and H. Prade, "Rough fuzzy sets and fuzzy rough sets," *International Journal of General Systems*, vol. 17, no. 2-3, pp. 191–209, 1990.

[15] W. Wu, J. Mi, and W. Zhang, "Generalized fuzzy rough sets," *Information Sciences*, vol. 151, pp. 263–282, 2003.

[16] W.-Z. Wu and W.-X. Zhang, "Constructive and axiomatic approaches of fuzzy approximation operators," *Information Sciences*, vol. 159, no. 3-4, pp. 233–254, 2004.

[17] S. Greco, B. Matarazzo, and R. Slowinski, "Rough approximation by dominance relations," *International Journal of Intelligent Systems*, vol. 17, no. 2, pp. 153–171, 2002.

[18] R. Slowinski and D. Vanderpooten, "A generalized definition of rough approximations based on similarity," *IEEE Transactions on Knowledge and Data Engineering*, vol. 12, no. 2, pp. 331–336, 2000.

[19] Y. Y. Yao, "Constructive and algebraic methods of the theory of rough sets," *Information Sciences*, vol. 109, no. 1–4, pp. 21–47, 1998.

[20] D. Boixader, J. Jacas, and J. Recasens, "Upper and lower approximations of fuzzy sets," *International Journal of General Systems*, vol. 29, no. 4, pp. 555–568, 2000.

[21] W. Wu and W. Zhang, "Neighborhood operator systems and approximations," *Information Sciences*, vol. 144, no. 1–4, pp. 201–217, 2002.

[22] C. Degang, W. Changzhong, and H. Qinghua, "A new approach to attribute reduction of consistent and inconsistent covering decision systems with covering rough sets," *Information Sciences*, vol. 177, no. 17, pp. 3500–3518, 2007.

[23] W. Zhu and F. Wang, "On three types of covering-based rough sets," *IEEE Transactions on Knowledge and Data Engineering*, vol. 19, no. 8, pp. 1131–1143, 2007.

[24] L. A. Zadeh, "Toward a theory of fuzzy information granulation and its centrality in human reasoning and fuzzy logic," *Fuzzy Sets and Systems*, vol. 90, no. 2, pp. 111–127, 1997.

[25] Y. H. Qian and J. Y. Liang, "Rough set method based on multigranulations," in *Proceedings of the 5th IEEE International Conference on Cognitive Informatics (ICCI '06)*, pp. 297–304, Beijing, China, July 2006.

[26] W. H. Xu, X. T. Zhang, and Q. R. Wang, "A generalized multigranulation rough set approach," *Lecture Notes in Computer Science*, vol. 6840, pp. 681–689, 2012.

[27] W. H. Xu, Q. R. Wang, and X. T. Zhang, "Multi-granulation fuzzy rough sets in a fuzzy tolerance approximation space," *International Journal of Fuzzy Systems*, vol. 13, no. 4, pp. 246–259, 2011.

[28] X. Wei-Hua, Z. Xiao-Yan, and Z. Wen-Xiu, "Knowledge granulation, knowledge entropy and knowledge uncertainty measure in ordered information systems," *Applied Soft Computing*, vol. 9, no. 4, pp. 1244–1251, 2009.

[29] W. Xu, W. Sun, X. Zhang, and W. Zhang, "Multiple granulation rough set approach to ordered information systems," *International Journal of General Systems*, vol. 41, no. 5, pp. 475–501, 2012.

[30] X. B. Yang, Y. S. Qi, X. N. Song, and J. Y. Yang, "Test cost sensitive multigranulation rough set: model and minimal cost selection," *Information Sciences*, vol. 250, pp. 184–199, 2013.

[31] X. B. Yang, Y. H. Qian, and J. Y. Yang, "Hierarchical structures on multigranulation spaces," *Journal of Computer Science and Technology*, vol. 27, no. 6, pp. 1169–1183, 2012.

[32] X. B. Yang, Y. H. Qian, and J. Y. Yang, "On characterizing hierarchies of granulation structures via distances," *Fundamenta Informaticae*, vol. 123, no. 3, pp. 365–380, 2013.

[33] X. Yang, X. Song, Y. She, and J. Yang, "Hierarchy on multigranulation structures: a knowledge distance approach," *International Journal of General Systems*, vol. 42, no. 7, pp. 754–773, 2013.

[34] X. B. Yang, X. N. Song, Z. H. Chen, and J. Y. Yang, "Multigranulation rough sets in incomplete information system," in *Incomplete Information System and Rough Set Theory*, pp. 195–222, Springer, Berlin, Germany, 2012.

[35] G. Lin, Y. Qian, and J. Li, "NMGRS: neighborhood-based multigranulation rough sets," *International Journal of Approximate Reasoning*, vol. 53, no. 7, pp. 1080–1093, 2012.

[36] Y. H. Qian, H. Zhang, Y. L. Sang, and J. Y. Liang, "Multigranulation decision-theoretic rough sets," *International Journal of Approximate Reasoning*, vol. 55, pp. 225–237, 2014.

[37] Y. H. She and X. L. He, "On the structure of the multigranulation rough set model," *Knowledge-Based Systems*, vol. 36, pp. 81–92, 2012.

[38] W. Xu, W. Sun, and Y. Liu, "Optimistic multi-granulation fuzzy rough set model based on triangular norm," *Lecture Notes in Computer Science*, vol. 7414, pp. 20–27, 2012.

[39] W. H. Xu, W. X. Sun, and Y. F. Liu, "Optimistic multi-granulation fuzzy rough set model in ordered information system," in *Proceedings of the IEEE International Conference on Granular Computing (GrC '12)*, pp. 563–568, Hangzhou, China, August 2012.

[40] A. P. Dempster, "Upper and lower probabilities induced by a multivalued mapping," *Annals of Mathematical Statistics*, vol. 38, pp. 325–339, 1967.

[41] G. Shafer, *A Mathematical Theory of Evidence*, Princeton University Press, Princeton, NJ, USA, 1976.

[42] L. A. Zadeh, "Fuzzy sets and information granularity," in *Advances in Fuzzy Set Theory and Applications*, M. Gupta, R. Ragade, and R. R. Yager, Eds., pp. 3–18, North-Holland, Amsterdam, The Netherlands, 1979.

[43] L. A. Zadeh, "Fuzzy sets as a basis for a theory of possibility," *Fuzzy Sets and Systems*, vol. 1, no. 1, pp. 3–28, 1978.

[44] W. Wu, Y. Leung, and J. Mi, "On generalized fuzzy belief functions in infinite spaces," *IEEE Transactions on Fuzzy Systems*, vol. 17, no. 2, pp. 385–397, 2009.

[45] A. Skowron and J. Graymala-Busse, "From rough set theory to evidence theory," in *Advance in the Dempster-Shafer Theory of Evidence*, RR. Yager, M. Fedrizzi, and J. Kacprzyk, Eds., pp. 193–236, Wiley, New York, NY, USA, 1994.

[46] W. Z. Wu, Y. Leung, and W. X. Zhang, "Connections between rough set theory and Dempster-Shafer theory of evidence," *International Journal of General Systems*, vol. 31, no. 4, pp. 405–430, 2002.

[47] W. Z. Wu, M. Zhang, H. Z. Li, and J. S. Mi, "Knowledge reduction in random information systems via Dempster-Shafer theory of evidence," *Information Sciences*, vol. 174, no. 3-4, pp. 143–164, 2005.

[48] Y. Y. Yao and P. J. Lingras, "Interpretations of belief functions in the theory of rough sets," *Information Sciences*, vol. 104, no. 1-2, pp. 81–106, 1998.

[49] X. B. Yang and J. Y. Yang, *Incomplete Information System and Rough Set Theory: Models an d Attribute Reductions*, Science Press, Beijing, China, 2012.

[50] W. X. Zhang, J. S. Mi, and W. Z. Wu, "Approaches to knowledge reductions in inconsistent systems," *International Journal of Intelligent Systems*, vol. 21, no. 9, pp. 989–1000, 2003.

[51] W. X. Zhang, Y. Leung, and W. Z. Wu, *Information Systems and Knowledge Discovery*, Science Press, Beijing, China, 2004.

[52] M. Kryszkiewicz, "Comparative studies of alternative type of knowledge reduction in inconsistent systems," *International Journal of Intelligent Systems*, vol. 16, pp. 105–120, 2001.

[53] Y. Leung, W.-Z. Wu, and W.-X. Zhang, "Knowledge acquisition in incomplete information systems: a rough set approach," *European Journal of Operational Research*, vol. 168, no. 1, pp. 164–180, 2006.

[54] W.-H. Xu and W.-X. Zhang, "Measuring roughness of generalized rough sets induced by a covering," *Fuzzy Sets and Systems*, vol. 158, no. 22, pp. 2443–2455, 2007.

[55] W. X. Zhang, W. Z. Wu, J. Y. Liang, and D. Y. Li, *Theory and Method of Rough Sets*, Science Press, Beijing, China, 2001.

[56] Y. Leung and D. Li, "Maximal consistent block technique for rule acquisition in incomplete information systems," *Information Sciences*, vol. 153, pp. 85–106, 2003.

[57] J. Y. Liang and D. Y. Li, *Uncertainty and Knowledge Acquisition in Information Systems*, Science Press, Beijing, China, 2005.

[58] Y.-H. Qian, J.-Y. Liang, Y.-Y. Yao, and C.-H. Dang, "MGRS: a multi-granulation rough set," *Information Sciences*, vol. 180, no. 6, pp. 949–970, 2010.

[59] W. H. Xu, X. Y. Zhang, and W. X. Zhang, "Two new types of multi-granulation rough set," submitted to *Information Sciences*.

[60] W. X. Zhang, Y. Leung, and W. Z. Wu, "Fuzzy information systems and knowledge discovery," in *Information Systems and Knowledge Discovery*, pp. 96–112, Science Press, 2003.

[61] Y. H. Qian, J. Y. Liang, W. Pedrycz, and C. Y. Dang, "Positive approximation: an accelerator for attribute reduction in rough set theory," *Artificial Intelligence*, vol. 174, no. 9-10, pp. 597–618, 2010.

[62] Y. H. Qian, J. Y. Liang, and W. Wei, "Pessimistic rough decision," in *Proceedings of the 2nd International Workshop on Rough Sets Theory*, pp. 440–449, 2010.

Hindawi Publishing Corporation
Journal of Applied Mathematics
Volume 2014, Article ID 742341, 9 pages
http://dx.doi.org/10.1155/2014/742341

Research Article

Precomputed Clustering for Movie Recommendation System in Real Time

Bo Li,[1] Yibin Liao,[1] and Zheng Qin[2]

[1] *Department of Computer Science, University of Georgia, Athens, GA 30602, USA*
[2] *School of Software, Tsinghua University, Beijing 100084, China*

Correspondence should be addressed to Zheng Qin; qingzh@tsinghua.edu.cn

Received 7 February 2014; Accepted 12 May 2014; Published 25 June 2014

Academic Editor: Guiming Luo

A recommendation system delivers customized data (articles, news, images, music, movies, etc.) to its users. As the interest of recommendation systems grows, we started working on the movie recommendation systems. Most research efforts in the fields of movie recommendation system are focusing on discovering the most relevant features from users, or seeking out users who share same tastes as that of the given user as well as recommending the movies according to the liking of these sought users or seeking out users who share a connection with other people (friends, classmates, colleagues, etc.) and make recommendations based on those related people's tastes. However, little research has focused on recommending movies based on the movie's features. In this paper, we present a novel idea that applies machine learning techniques to construct a cluster for the movie by implementing a distance matrix based on the movie features and then make movie recommendation in real time. We implement some different clustering methods and evaluate their performance in a real movie forum website owned by one of our authors. This idea can also be used in other types of recommendation systems such as music, news, and articles.

1. Introduction

Movie recommendation systems suggest movies to user that he/she might be interested in. The generated suggestions are obtained from the consideration of many aspects. The suggestion can be based on the tastes, interests, goals, people connections, and so forth.

In general, a movie recommendation system compares user's profile or usage data to some reference characteristics and combines the user's social environment to make movie recommendations. This type of recommendations is based on user. However, this type of recommendations may not work or make inaccurate recommendation in the following situations. The user does not have strong profile setting in the system. There are many users who do not want to set their profile due to laziness or privacy concerns. In this case, most recommendation systems consider the user's social connection in the system such as friends, classmates, families, and colleagues. However, the tastes of the movies may be various even among best friends. What is worse, the "friends" that the user has added in the website may not be people with the same interest. For example, in a community network or local network such as a university or college, like our experimental environment, users' social connections are built mainly because they are in the same university with the same major and similar age; however, their tastes towards movies may be totally different, which fails the fundamental bias in the recommendation system mentioned above.

Because of the situation stated above, we propose the use of machine learning (ML) techniques (clustering) to analyze the movie features and system logs (user's voting logs) to make correct recommendations more adequately. In this proposed approach, we compute a distance matrix for the movie features and apply the clustering techniques to classify movies into different areas off-line. For every user logged on the system, we recommend movies from the clusters combined with the user's majority voting result in real time.

In order to compare the accuracy and efficiency, we have implemented different clustering techniques as follows:

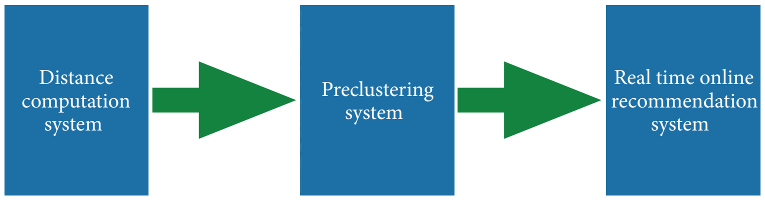

Distance computation system
Preclustering system
Real time online recommendation system

Figure 1: System overview.

DBSCAN (density-based clustering), affinity propagation, hierarchical clustering, and random clustering as a base line.

We have tested this proposal with all the clustering techniques in a university social website owned by one of our authors. There is a subsystem with movie service in the website. We add a recommendation function in a banner and put it on top of the section to let user make votes. We offer them all the clustering results from four cluster algorithms shuffled together and record the voting results separately.

The rest of the paper is organized as follows. Section 2 gives an overview of related works in which machine learning techniques have been applied to recommendation systems. Section 3 is the system overview of our recommendation system. Section 4 illuminates the distance functions we use to compute the distance matrix. Section 5 introduces all the clustering techniques used in our preclustering system. Section 6 includes the experimental setup and evaluation results. Finally, Section 7 concludes with discussions and future research.

2. Related Work

Machine learning techniques are very useful when huge amounts of data have to be classified and analyzed, which nowadays is a very common situation in many scenarios, especially in the recommendation system. There are a lot of different machine learning techniques used in different recommendation systems, such as Naive Bayes classification [1], decision tree [2], k-means clustering and improvement [3–6], and so forth.

Generally, the recommendation systems are divided into two major categories such as collaborative recommendation system and content based recommendation system [7]. In case of collaborative recommendation systems, these try to seek out users who share same tastes as that of the given user as well as to recommend the movies according to the liking of these sought users. For example, Tatli and Birtürk described an approach for creating music recommendations based on user-supplied tags that are augmented with a hierarchical structure extracted for top level genres from Dbpedia. In this structure, each genre is represented by its stylistic origins, typical instruments, derivative forms, subgenres, and fusion genres. They use this well-organized structure in dimensionality reduction in user profiling [8]. Yang et al. proposed a personalized web page recommendation model called PIGEON (abbreviation for PersonalIzed web paGe rEcommendatiON) via collaborative filtering and a topic-aware Markov model and proposed a graph-based iteration algorithm to discover user's interested topics, based on which user similarities are measured [9]. Cai et al. also proposed a model that fully captures the bilateral role of user interactions within a social network and formulated collaborative filtering methods to enable people to people recommendation [10]. As I mentioned in Section 1, making recommendation based on user is not working properly in some particular areas such as a community network system or university network.

The content based recommendation systems try to recommend contents similar to those web sites the user has liked. Biancalana et al. proposed two different context-aware approaches for the movie recommendation task, one is a hybrid recommender that assesses available contextual factors related to time in order to increase the performance of traditional CF approaches, and the other one aims at identifying users in a household that submitted a given rating [11]. This latter approach is based on machine learning techniques, namely, neural networks and majority voting. In our paper, we focus on the content based recommendation and use the majority voting and clustering techniques. We obtained better results compared to their research, which will be shown in Section 6.

Some of the researchers proposed hybrid recommendation approaches by combining different approaches. Ghazanfar and Prügel-Bennett proposed a unique switching hybrid recommendation approach by combining a Naive Bayes classification approach with the collaborative filtering [1]. Ujwala et al. presented and investigated an approach based on weighted Association Rule Mining Algorithm and text mining [7]. Bellogín et al. implemented decision trees and attribute selections together to build the recommendation model to find the most relevant preferences of user and system [2]. The idea of hybrid recommendation approach gives us inspiration to combine different approaches to implement the movie recommendation for future work.

3. System Overview

Figure 1 provides a high-level overview of our new movie recommendation approach. As shown in Figure 1, the new movie recommendation system comes in three parts: distance computation system, preclustering system, and real time online recommendation system.

3.1. Distance Computation System. Distance computation system computes the distance between different movies based

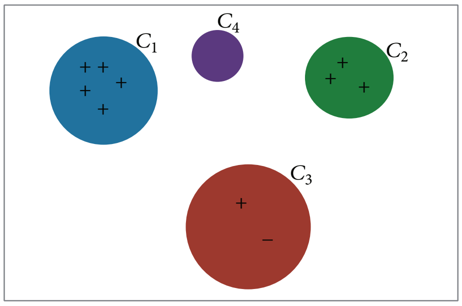


Figure 2: A demo of voting. The probability that recommended movies are in C_1 is 5/8.

on different properties of the movies including the movies types, publish year, countries of publishing companies, languages, directors, casts, and duration time. We first compute the Jaccard distance based on movies types, countries of publishing companies, languages, directors, and casts, and then the distance based on publish year and duration time by the distance we defined in Sections 4.2.2 and 4.2.3. We then compute the overall distances between each movie sample by summarizing them together with the weights which we obtained from the survey mentioned in Section 4.1.

3.2. Preclustering System. Preclustering system separates movies into different clusters before giving them to online recommendation system. In this system, we tried five kinds of different clustering algorithms: affinity propagation, DBSCAN, hierarchical clustering, spectral clustering, and random generator, among which spectral clustering is too slow to get the result in demanded time. Therefore, we sent the remaining four clustering results to the recommendation system with two data formats, one can get the cluster label by one movie's IMDB identifier and another one can drag all the movies' IMDB identifiers by one of cluster labels. In this way, the online recommendation system in Section 3.3 can quickly get the information it wants.

3.3. Real Time Online Recommendation System. After getting the preclustering results, the website read the history of users' votes on movies and used the majority of the voting to generate recommendations. In more detail, if one user U_i likes movie M_{j1} which belongs to cluster C_{k1}, U_i will give C_{k1} one *vote* on C_{k1}. On contract, if U_i dislikes movie M_{j2} in C_{k2}, U_i will cancel a *vote* for C_{k2}. If a cluster gets negative *vote* at the end, it will return to 0. After U_i votes all the movies he/she comments, different clusters will get different votes. The movie recommendation system will recommend movies according to the votes of clusters C_j as $V(C_j)$ as follows:

$$P(C_i) = \frac{V(C_i)}{\sum_{j=1}^{n} V(C_j)}, \tag{1}$$

where $P(C_i)$ is the probability that a movie should be recommended from cluster C_i and $V(C_i)$ is the number of votes in cluster C_i. n is the number of clusters. As shown in Figure 2.

To make sure that in our evaluation the reason that user votes a movie we recommended as dislike is not because the movie itself is really bad, we recommend movies with the IMDB score higher than 7 in the clusters which have already been voted out by the user.

4. Distance Computation System

4.1. Weight Survey. Movies' properties create a very special space, where the weights of each dimension are treated completely different. For example, the type of a movie is of course more important than the duration of it. Therefore, we cannot use the default distance function provided by Scikit-learn [18] package that we use. Instead, the feature selection like in [2] or starting a survey to figure out the weights to different dimensions should be implemented. In other research areas where we human beings do not know which feature is important, such as features in carcinogenic, we always use feature selection to decide the weights of different features. In movie recommendation area, however, we can investigate it by the survey shown in Figure 3, since people know why they like the movie. By carefully designing a survey and letting users vote the top three factors, we obtained a result shown in Figure 4.

From the survey result shown in Figure 4, we can figure out that the weight of publish year is 0.0915, weight of country is 0.147, weight of type is 0.4167, weight of language is 0.0545, weight of director is 0.0896, weight of casts is 0.1792, and weight of duration is 0.0214. These weights will be used in the overall distance computation in Section 4.3.

4.2. Distance Function

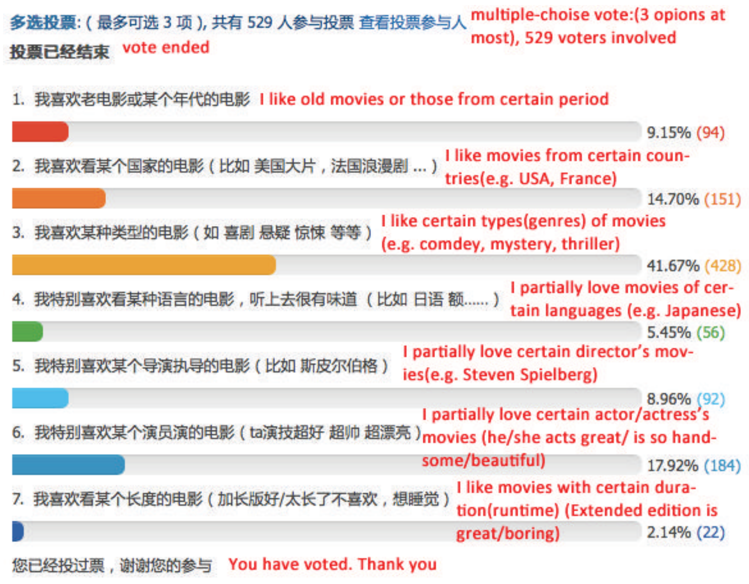
4.2.1. Jaccard Distance. Since for countries, languages, casts, and types of the movie, the distance is determined by how many same and different items there are between sets, we decided to use Jaccard distance proposed in [12]. Jaccard distance is a statistic used for measuring dissimilarity between sample sets. The formula is shown in formula (2). According to this formula, we use intersection and union to do the calculation and easily obtain the distance between each movie feature mentioned above as follows:

$$J(A,B) = 1 - \frac{|A \cap B|}{|A \cup B|}. \tag{2}$$

4.2.2. Publish Year Distance. The distance for publish year between movies is a very special case for the human being. To human's intuition, the distance between movies published in 1950s and 1960s seems much less than that between movies published in 2012 and 2013. In another word, the distance of publish year is not a linear function. One reason for such phenomenon is that there are more and more movies published recently. The statistic of number of movies in each year in our database is shown in Figure 5.

Figure 5 illuminates that the number of published movies in each year is similar to an exponential function. The trend line is shown as follows:

$$f(x) = 0.3584 \cdot \exp^{0.0598(x-1902)}. \tag{3}$$



多选投票: (最多可选 3 项), 共有 529 人参与投票 查看投票参与人 multiple-choise vote:(3 opions at most), 529 voters involved

投票已经结束 vote ended

1. 我喜欢老电影或某个年代的电影 I like old movies or those from certain period
   9.15% (94)
2. 我喜欢看某个国家的电影 (比如 美国大片，法国浪漫剧 ...) I like movies from certain countries(e.g. USA, France)
   14.70% (151)
3. 我喜欢某种类型的电影 (如 喜剧 悬疑 惊悚 等等) I like certain types(genres) of movies (e.g. comdey, mystery, thriller)
   41.67% (428)
4. 我特别喜欢看某种语言的电影，听上去很有味道 （比如 日语 额...... ） I partially love movies of certain languages (e.g. Japanese)
   5.45% (56)
5. 我特别喜欢某个导演执导的电影 (比如 斯皮尔伯格) I partially love certain director's movies(e.g. Steven Spielberg)
   8.96% (92)
6. 我特别喜欢某个演员演的电影 (ta演技超好 超帅 超漂亮) I partially love certain actor/actress's movies (he/she acts great/ is so handsome/beautiful)
   17.92% (184)
7. 我喜欢看某个长度的电影 (加长版好/太长了不喜欢，想睡觉) I like movies with certain duration(runtime) (Extended edition is great/boring)
   2.14% (22)

您已经投过票，谢谢您的参与 You have voted. Thank you

Figure 3: Survey in our community.

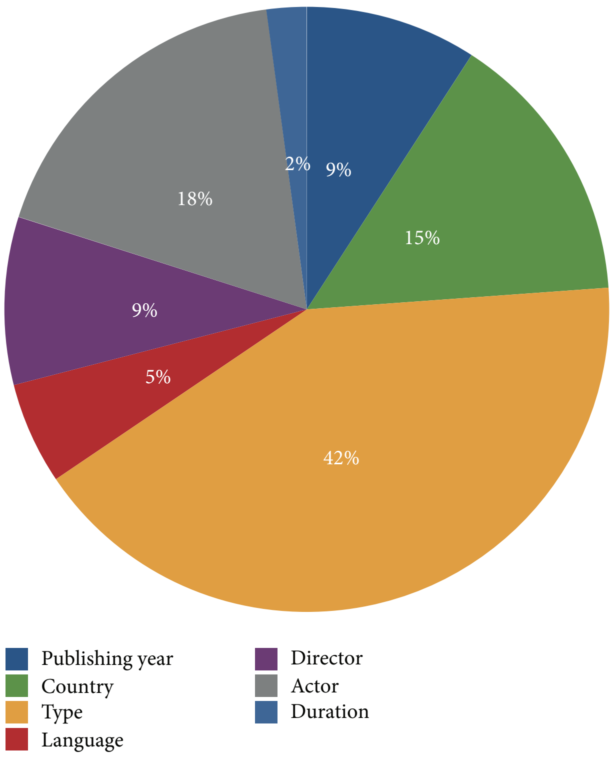


Figure 4: Survey of top three liked movie factors.

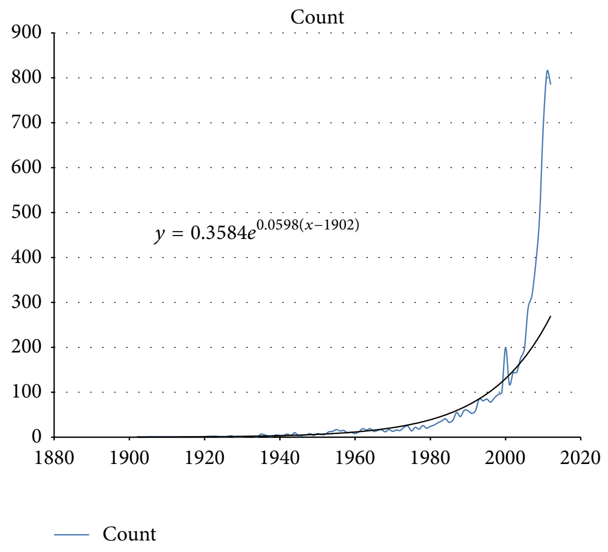


Figure 5: Statistic result of movie published in each year.

According to formula (3), we propose a new distance function for publish year $\mathrm{PD}(\mathrm{year}_i, \mathrm{year}_j)$ shown in the following:

$$\mathrm{PD}\left(\mathrm{year}_i, \mathrm{year}_j\right) = \frac{\int_{\min(\mathrm{year}_i,\mathrm{year}_j)}^{\max(\mathrm{year}_i,\mathrm{year}_j)} f(x)}{\int_{1902}^{2014} f(x)}. \quad (4)$$

4.2.3. Duration Distance. Human being is very sensitive for a small period of time, which means that half an hour is nearly as double time as a quarter of an hour in human's feeling. Therefore, we use a linear function shown in formula (5) to compute the duration distance $\mathrm{DUD}(\mathrm{dur}_i, \mathrm{dur}_j)$

$$\mathrm{DUD}\left(\mathrm{dur}_i, \mathrm{dur}_j\right) = \frac{\min\left(\left|\mathrm{dur}_i - \mathrm{dur}_j\right|, 200\right)}{200}. \quad (5)$$

Here, in order to normalize the distance, we consider 200 minutes as the maxima duration of a movie.

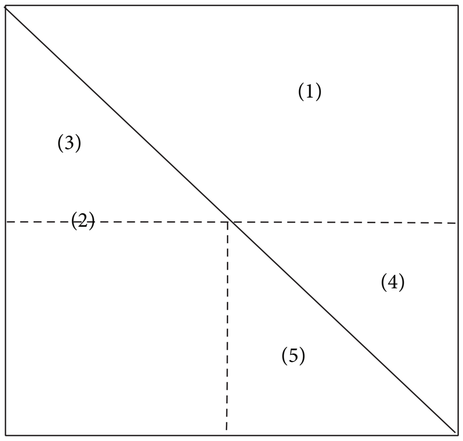


Figure 6: Matrix computation parallel demo.

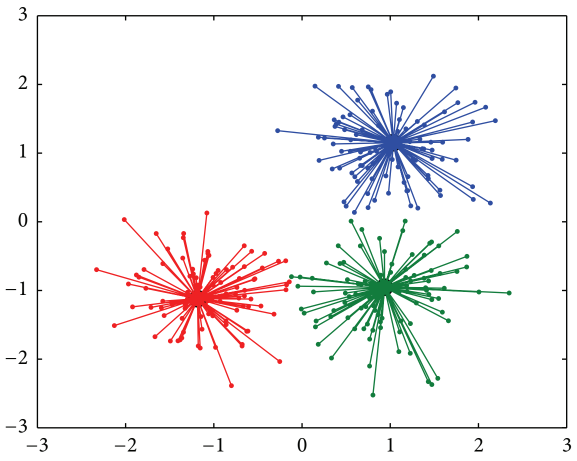


Figure 7: Affinity propagation demo [18].

4.3. Overall Distance. We build up a distance matrix $[n, n]$ between each movie in our database based on formula (6) as follows:

$$\mathrm{OD}\left(m_i, m_j\right) = \sum_{k \in \{\text{movie features}\}} \text{weight}_k * \text{distance}_k\left(i, j\right). \quad (6)$$

4.4. Parallel Distance Computation. Since our distance matrix is symmetric, we can use such property to do distance computation in parallel.

To do parallel computation, two conditions must be satisfied.

(1) Works assigned to different processing must be approximately equal.

(2) There is no interaction or data exchange between different working processing.

The distance matrix we need to build is a symmetric matrix in which all the elements on the main diagonal are equal to 0, formally defined as a matrix D, where $D = D^T$ and $D[i, i] = 0$, as follows:

$$D = \begin{bmatrix} 0 & \lambda_{0,1} & \lambda_{0,2} & \cdots & \lambda_{0,n-1} & \lambda_{0,n} \\ \lambda_{0,1} & 0 & \lambda_{1,2} & \cdots & \lambda_{1,n-1} & \lambda_{1,n} \\ \lambda_{0,2} & \lambda_{1,2} & 0 & \cdots & \lambda_{2,n-1} & \lambda_{2,n} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ \lambda_{0,n-1} & \lambda_{1,n-1} & \lambda_{2,n-1} & \cdots & 0 & \lambda_{n-1,n} \\ \lambda_{0,n} & \lambda_{1,n} & \lambda_{2,n} & \cdots & \lambda_{n-1,n} & 0 \end{bmatrix}. \quad (7)$$

The scratch of the distance matrix shown in formula (7) is shown in Figure 6. Since the distance matrix is symmetric, the data in area (1) and area (2) and that in area (4) and area (5) are correspondingly equal. Therefore instead of computing all of them, we just compute the upper triangle of the matrix, that is, the data in areas (1) and (5). However, if we separate the parallel tasks by rows of the matrix, the works (number of elements) in each task are (is) not equal (from $n - 1$ to 1). To overcome this problem, the works needed in area (4) are filled into area (3), so that we can group $n - 1$ unequal tasks into $\lceil n/2 \rceil$ equal tasks. In this way, a process pool where each process computes distance in one task can be used to parallel these $\lceil n/2 \rceil$ tasks.

More specifically, we firstly assign the main diagonal line to 0. There is no need to compute the main diagonal, since we know that the distance between one movie and itself is 0. Then for each row i, we combine the distance computations of $D[n-1-i, n-i]$, $D[n-1-i, n-i+1]$,. . .,$D[n-1-i, n-2]$, $D[n-1-i, n-1]$, and those of $D[i, i+1]$, $D[i, i+2]$,. . .,$D[i, n-2]$, $D[i, n-1]$ together as a task. After computation, we get an upper triangular matrix, that is, area (1) and area (4). Then fill the lower triangle with the upper triangle; that is, let $D = D + D^T$.

5. Preclustering System

In this section, we use four clustering methods to investigate the best cluster method which can be used in the final movie recommendation system. The four clustering algorithms are affinity propagation [13], DBSCAN [14], hierarchical clustering [15], and random clustering for base line usage.

5.1. Clustering Method

5.1.1. Affinity Propagation. Affinity propagation which was first proposed in [13] generates clusters by sending messages between pairs of samples until convergence or changes falling below a threshold. A dataset is then described using a small number of exemplars, which are identified as those most representative of other samples. The messages sent between pairs represent the suitability for one sample to be the exemplar of the other, which is updated in response to the values from other pairs. This updating happens iteratively until convergence, at which point the final exemplars are chosen, and hence the final clustering is given as demonstrated in Figure 7.

5.1.2. DBSCAN. The DBSCAN algorithm [14] views clusters as areas of high density separated by areas of low density.

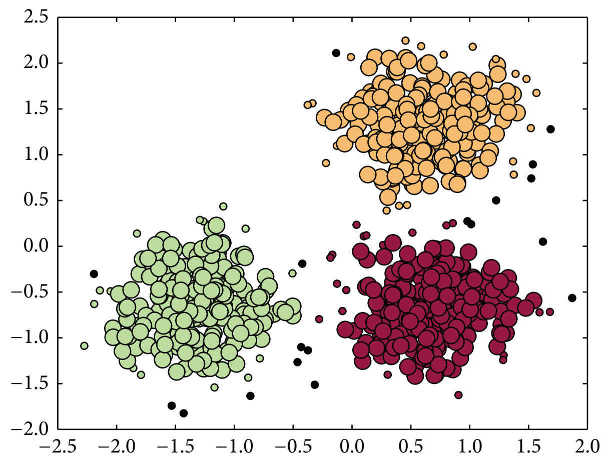


Figure 8: DBSCAN demo [18].

Due to this rather generic view, clusters found by DBSCAN can be in any shape, as opposed to k-means which assumes that clusters are convex shaped. The central component to the DBSCAN is the concept of core samples, which are samples that are in areas of high density. A cluster is therefore a set of core samples, each close to each other (measured by some distance measure) and a set of non-core samples that are close to a core sample (but are not themselves core samples) as demonstrated in Figure 8.

5.1.3. Hierarchical Clustering. Hierarchical clustering [15] is a general family of clustering algorithms that build nested clusters by merging them successively. This hierarchy of clusters is represented as a tree. The root of the tree is the unique cluster that gathers all the samples, the leaves being the clusters with only one sample.

The beauty of this clustering method is that we have no need to provide the size of clustering ahead. Instead, we can initialize with a large cut height and decrease the cut height when the user has voted most movies in one cluster. In this way, we can provide a larger cluster to fit user's interests. As shown in Figure 9.

5.1.4. Random Clustering. We first fix the size of clustering to 60 and then for each movie, we randomly pick the cluster it belongs to by generating a random number between 1 and 60.

5.2. Preclustering System Output. Because of seeking efficiency, we output the clustering label results in two formats. One can be used to check clustering labels by movie ID and another is used to check all the movies' IDs in one cluster given the cluster label. By outputting the file in JSON, we can easily transfer data between preclustering system in Python and real time online recommendation system in PHP.

6. Evaluation

Before the evaluation starts, we created a survey and asked users to select the top three movie features they care about as mentioned in Section 4.1. After one week of data collection, we have the result which is shown in Figure 4 as mentioned.

Table 1: Comparison of results between different clustering methods in first evaluation method.

	True positive	False positive	Accuracy rate
AF	239	98	70.920%
DB	174	77	69.323%
HC	**228**	**88**	**72.152%**
RA	125	104	54.585%

Based on the result, we calculate the distance matrix and make it as an input to the different clusters. We implemented the four clustering techniques by using python library Scikit-learn.

6.1. Experimental Setup. The evaluation has been conducted using a university social website in China since the website is owned by one of our authors. We add the recommendation function in a banner and put it on top of the movie service to let users vote. The overview voting component of the website is showed in Figure 10.

There are three voting choices: like, do not like, and unseen for each recommended movie. Voting like scores 1, voting do not like scores −1, and voting unseen scores 0. Because of the limited time of data collection, we cannot wait for users to watch the recommended movies. Therefore, we add the unseen choices for users to select if they have not seen the recommended movies and only consider the votes which are like or do not like to evaluate our approach.

6.2. Result. After two weeks voting for the movie recommendation, we have collected 168,424 total votes, 132,360 votes are collected in 12 days and used for recommendation majority voting, as mentioned in Section 3.3. The remaining 6064 votes are collected in two days after the recommendation system is deployed. Since only the like vote and do not like vote are considered illegible, we generated the results in the following subsections.

6.2.1. First Evaluation Result. In the first evaluation result, listed in Table 1, we calculated the true positive, false positive, and accuracy for each clustering techniques. True positive equals the number of like votes, and false positive equals the number of do not like votes. Because most of the recommended movies are voted unseen, there are a total of 1,133 votes counted, and 216 users participated in the voting. Consider the following:

$$\text{Accuracy} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Positive}}. \quad (8)$$

As shown by the result in Table 1, the hierarchical clustering approach is the best. However, the accuracy of every clustering approach is not satisfied. Therefore, we extracted the users' voting history from our server database and analyzed those data. We found that some of the users made less than 5 votes, some of them made between 5 to 10 votes.

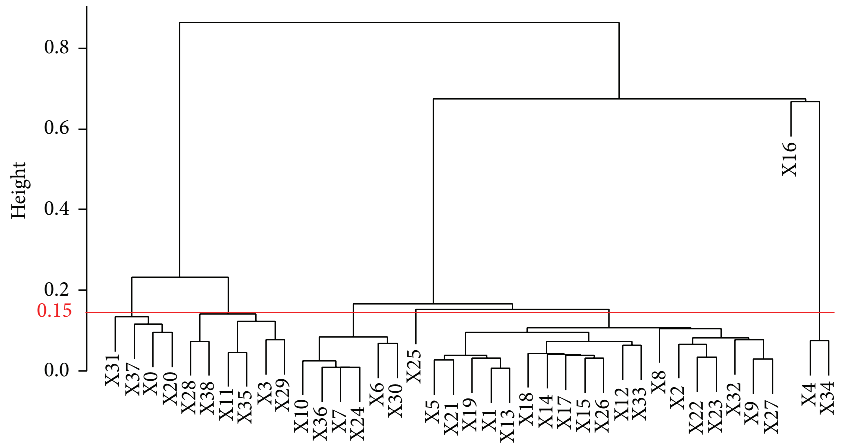


Figure 9: Hierarchical clustering demo.

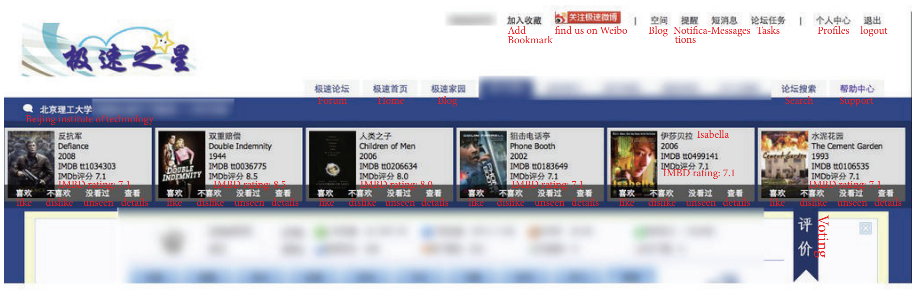

极速之星
加入收藏 Add Bookmark
关注极速微博 find us on Weibo
空间 Blog
提醒 Notifica-tions
短消息 Messages
论坛任务 Tasks
个人中心 Profiles
退出 logout
极速论坛 Forum
极速首页 Home
极速家园 Blog
论坛搜索 Search
帮助中心 Support
北京理工大学 Beijing institute of technology
反抗军 Defiance 2008 IMDB tt1034303 IMDb评分 7.1 IMBD rating: 7.1
双重赔偿 Double Indemnity 1944 IMDB tt0036775 IMDb评分 8.5 IMBD rating: 8.5
人类之子 Children of Men 2006 IMDB tt0206634 IMDb评分 8.0 IMBD rating: 8.0
狙击电话亭 Phone Booth 2002 IMDB tt0183649 IMDb评分 7.1 IMBD rating: 7.1
伊莎贝拉 Isabella 2006 IMDB tt0499141 IMDb评分 7.1 IMBD rating: 7.1
水泥花园 The Cement Garden 1993 IMDB tt0106535 IMDb评分 7.1 IMBD rating: 7.1
喜欢 like
不喜欢 dislike
没看过 unseen
查看 details
评价 Voting

Figure 10: Website overview.

There are a few users who made more than 20 votes. In those people who made more than 20 votes, we found that 4 users made all their votes do not like. Therefore, almost one third of the false positive votes were made by only 4 users over 216 users. Table 3 shows one example from one typical user. The first column is the movie ID from imdb, the second column is the voting results, and the third column is the voting time. We think maybe this user dislikes our recommendation idea so he made all the votes negative.

There is another small portion of users made more than 20 votes. Most of the votes are negative. Only a few votes are positive, and no zero or unseen votes. Table 4 shows one of such examples. It is probably because this user is voting negative instead of the unseen choices for the unseen movies, since most of the users were voting unseen more than like and do not like.

6.2.2. Second Evaluation Result. These two situations mentioned in Section 6.2.1 effect our result a lot in the first evaluation. Therefore, we made a second evaluation by calculating each user's accuracy in formula (8) and then computing the average of all user's accuracy. Consider the following:

$$\text{Accuracy} = \frac{\sum \text{accuracy}}{\text{Total Number of Users}}. \qquad (9)$$

Table 2: Comparison of results between different clustering methods in second evaluation method.

	Accuracy rate
AF	80.95%
DB	**84.71%**
HC	82.03%
RA	63.51%

In the second evaluation, we wait for another 6 hours to collect more data. There are 7,100 votes in total, 1420 votes are considered and 242 users participated in voting. Based on the second evaluation, we obtained much better result as shown in Table 2. DBSCAN (density-based clustering), affinity propagation, and hierarchical clustering obtained over 80% accuracy. DBSCAN reached to 84.71%.

6.3. Comparison with Related Work. In this section, we compared our result with those of with several other movie recommendation approach/systems. Pomerantz and Dudek [16] used a hierarchical Bayesian approach combined with the item similarity of the content for the movie recommender and achieved the classifiers of 70.8% as the best result. In Biancalana et al.'s paper [11], the best result is 82.4% obtained by combining three classifiers in a neural network. Basu et al.

Table 3: Example of all negative votes.

imdb	Vote	Last update
tt1716777	–1	2013-12-06 23:36:55
tt0183649	–1	2013-12-06 23:36:56
tt1728196	–1	2013-12-06 23:36:58
tt0014429	–1	2013-12-06 23:36:59
tt0034248	–1	2013-12-06 23:37:01
tt0017136	–1	2013-12-06 23:37:02
tt0035169	–1	2013-12-06 23:37:03
tt0023940	–1	2013-12-06 23:37:04
tt0035209	–1	2013-12-06 23:37:05
tt0047876	–1	2013-12-06 23:37:23
tt0106535	–1	2013-12-06 23:37:24
tt0121164	–1	2013-12-06 23:37:27
tt0006864	–1	2013-12-06 23:37:28
tt0499141	–1	2013-12-06 23:37:30
tt0049010	–1	2013-12-06 23:37:32
tt1298650	–1	2013-12-06 23:40:39
tt0408236	–1	2013-12-06 23:40:40
tt0162661	–1	2013-12-06 23:40:41
tt0349903	–1	2013-12-06 23:40:43
tt0114369	–1	2013-12-06 23:40:44
tt0096895	–1	2013-12-06 23:40:45
tt0111161	–1	2013-12-06 23:40:49
tt2992146	–1	2013-12-06 23:40:53
tt0082971	–1	2013-12-06 23:40:56
tt1075419	–1	2013-12-06 23:40:58

Table 4: Example of nonzero votes.

imdb	Vote	Last update
tt0068767	–1	2013-12-06 22:33:30
tt0109958	–1	2013-12-06 22:33:32
tt0047034	–1	2013-12-06 22:33:33
tt0397892	–1	2013-12-06 22:33:35
tt0032551	–1	2013-12-06 22:33:36
tt0110008	1	2013-12-06 22:33:38
tt0416881	–1	2013-12-06 22:33:40
tt1233499	–1	2013-12-06 22:33:42
tt1754691	–1	2013-12-06 22:33:45
tt1034303	–1	2013-12-06 22:33:47
tt0015324	–1	2013-12-06 22:33:48
tt0049408	–1	2013-12-06 22:33:50
tt0033467	–1	2013-12-06 22:33:51
tt0000417	–1	2013-12-06 22:33:54
tt1437358	–1	2013-12-06 22:33:56
tt0063105	–1	2013-12-06 22:33:57
tt0253474	–1	2013-12-06 22:33:58
tt0034248	1	2013-12-06 22:34:00
tt0018578	1	2013-12-06 22:34:04
tt0335345	–1	2013-12-06 22:34:07
tt0028445	1	2013-12-06 22:34:10
tt0040536	1	2013-12-06 22:34:16
tt0332379	–1	2013-12-06 22:34:19
tt1728196	–1	2013-12-06 22:34:21

[17] presented an inductive learning approach to recommendation and achieved an averaged precision of 83%.

Our best result (84.71%) achieved by DBSCAN [14] as shown in Table 2 improves 1.71% compared to best result of related works mentioned above. Meanwhile all these related works are computed offline and they use the stored models when testing, and these models cannot be updated in time since it takes too long to redo the learning process. Our approach, on the other hand, does not need to store user models and it updates the recommendation result in real time by preclustering movies and using a very simple learning approach (majority voting) to provide movie recommendation. Since movies are updated much less frequently than users' votes, our approach is much more time efficient than related works. To sum up, our research becomes significantly meaningful as it realizes the real time model updating while it does not compromise the accuracy.

7. Conclusion and Future Work

In this paper, we proposed a new distance function for publish year (year related) to compute the distance matrix, and then implemented a real time movie recommendation system based on content (movie) using preclustering and majority voting. We implemented four different clustering techniques in the preclustering process and obtained 84.71% accuracy as the best result, which is better than some of the other research papers' approaches. We realized a real time updating model with no compromises on accuracy. Our approach can even work better with special user groups such as people in colleges, universities, or professional communities.

The next step in this research is to make the system adaptive to be widely used by many other types of groups such as articles, news, and music, and to obtain better accuracy. One of the limitations in our approach is that if there is a cluster that contains a lot of positive votes as well as a lot of negative ones, we will not recommend movies in this cluster. However, this situation may happen because of two totally different reasons. One is that the user does not care about the movies in this cluster. In this case, no recommendation in this cluster is a wise choice. However, if the user really likes the movies in this cluster and he/she watched a lot of movies in this cluster, and some movies he/she likes while some he/she dislikes, ignoring this cluster, in such situation, will dissatisfy the user. To conquer this limitation, we can give weights to the votes from users. In the movie which is voted positive by the user and is voted positive by many other people as well, the weight of such vote will be decreased, while if the movie is voted positive by the user but is voted negative by most of others, this means such vote reveals the special taste from this user, and gains higher weight and vice versa. In a more specific way, we can set the weight of positive as $N/(P+N)$ and the weight of negative as $P/(P+N)$, where P is the total positive votes from all users towards this movie and N is the total negative votes from all users towards this movie. In this way, we can still ignore the cluster in first situation, while overcoming the limitation we had in the second situation.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

The authors especially thank Dr. Khaled Rasheed for providing them with suggestions and information regarding their research. The authors also thank Mr. Hongfei Yan for the related work and discussions.

References

[1] M. A. Ghazanfar and A. Prügel-Bennett, "An improved switching hybrid recommender system using Naive Bayes classifier and collaborative filtering," in *Proceedings of the International MultiConference of Engineers and Computer Scientists (IMECS '10)*, pp. 493–502, March 2010.

[2] A. Bellogín, I. Cantador, P. Castells, and A. Ortigosa, "Discovering relevant preferences in a personalised recommender system using machine learning techniques," in *Proceedings of the ECML-PKDD Workshop on Preference Learning*, 2008.

[3] F.-C. Lin, H.-W. Yu, C.-H. Hsu, and T.-C. Weng, "Recommendation system for localized products in vending machines," *Expert Systems with Applications*, vol. 38, no. 8, pp. 9129–9138, 2011.

[4] K.-J. Kim and H. Ahn, "A recommender system using GA K-means clustering in an online shopping market," *Expert Systems with Applications*, vol. 34, no. 2, pp. 1200–1209, 2008.

[5] S. B. Aher and L. M. R. J. Lobo, "Combination of machine learning algorithms for recommendation of courses in E-Learning System based on historical data," *Knowledge-Based Systems*, vol. 51, pp. 1–14, 2013.

[6] G. M. Dakhel and M. Mahdavi, "Providing an effective collaborative filtering algorithm based on distance measures and neighbors' voting," *International Journal of Computer Information Systems and Industrial Management Applications*, vol. 5, pp. 524–531, 2013.

[7] W. H. Ujwala, V. R. Sheetal, and M. Debajyoti, "A hybrid web recommendation system based on the improved association rule mining algorithm," *Journal of Software Engineering and Applications*, vol. 6, article 396, 2013.

[8] I. Tatli and A. Birtürk, "A tag-based hybrid music recommendation system using semantic relations and multi-domain information," in *Proceedings of the 11th IEEE International Conference on Data Mining Workshops (ICDMW '11)*, pp. 548–554, December 2011.

[9] Q. Yang, J. Fan, J. Wang, and L. Zhou, "Personalizing web page recommendation via collaborative filtering and topic-aware Markov model," in *Proceedings of the 10th IEEE International Conference on Data Mining (ICDM '10)*, pp. 1145–1150, December 2010.

[10] X. Cai, M. Bain, A. Krzywicki et al., "Collaborative filtering for people to people recommendation in social networks," in *AI 2010: Advances in Artificial Intelligence*, vol. 6464, pp. 476–485, Springer, Berlin, Germany, 2010.

[11] C. Biancalana, F. Gasparetti, A. Micarelli, A. Miola, and G. Sansonetti, "Context-aware movie recommendation based on signal processing and machine learning," in *Proceedings of the 2nd Challenge on Context-Aware Movie Recommendation*, pp. 5–10, October 2011.

[12] P. Jaccard, "The distribution of the flora in the alpine zone. 1," *The New Phytologist*, vol. 11, no. 2, pp. 37–50, 1912.

[13] B. J. Frey and D. Dueck, "Clustering by passing messages between data points," *Science*, vol. 315, no. 5814, pp. 972–976, 2007.

[14] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining (KDD '96)*, vol. 96, pp. 226–231, 1996.

[15] A. K. Jain and R. C. Dubes, *Algorithms for Clustering Data*, Prentice-Hall, Upper Saddle River, NJ, USA, 1988.

[16] D. Pomerantz and G. Dudek, "Context dependent movie recommendations using a hierarchical Bayesian model," in *Advances in artificial intelligence*, vol. 5549 of *Lecture Notes in Computer Science*, pp. 98–109, Springer, Berlin, Germany, 2009.

[17] C. Basu, H. Hirsh, and W. Cohen, "Recommendation as classification: using social and content-based information in recommendation," in *Proceedings of the 15th National Conference on Artificial Intelligence (AAAI '98)*, pp. 714–720, July 1998.

[18] "2.3. clustering—scikit-learn 0.14 documentation," http://scikit-learn.org/stable/modules/clustering.html#clustering.

Hindawi Publishing Corporation
Journal of Applied Mathematics
Volume 2014, Article ID 624863, 13 pages
http://dx.doi.org/10.1155/2014/624863

Research Article

Constitutive Relation of Engineering Material Based on SIR Model and HAM

Haoxiang He, Enzhen Han, and Maolin Cong

Beijing Laboratory of Earthquake Engineering and Structural Retrofit, Beijing University of Technology, Beijing 100124, China

Correspondence should be addressed to Haoxiang He; hhx7856@163.com

Received 7 March 2014; Revised 20 May 2014; Accepted 1 June 2014; Published 22 June 2014

Academic Editor: Xiaojing Yang

As an epidemic mathematical model, the SIR model represents the transition of the Susceptible, Infected, and Recovered. The profound implication of the SIR model is viewed as the propagation and dynamic evolutionary process of the different internal components and the characteristics in a complex system subject to external effect. The uniaxial stress-strain curve of engineering material represents the basic constitutive relation, which also represents the damage propagation in the units of the damaged member. Hence, a novel dynamic stress-strain model is established based on the SIR model. The analytical solution and the approximate solution for the proposed model are represented according to the homotopy analysis method (HAM), and the relationship of the solution and the size effect and the strain rate is discussed. In addition, an experiment on the size effect of confined concrete is carried out and the solution of SIR model is suitable for simulation. The results show that the mechanical mechanism of the parameters of the uniaxial stress-strain model proposed in this paper reflects the actual characteristics of the materials. The solution of the SIR model can fully and accurately show the change of the mechanical performance and the influence of the size effect and the strain rate.

1. Introduction

Engineering material means the material used for engineering or the materials used to produce other materials which may be used in engineering. The traditional material in civil engineering includes concrete, steel, soil, wood, and alloy. These materials have different qualities about strength, workability, durability, and resistance against corrosion [1]. The materials differ also in their structure, texture, and performance. The strength, toughness, and ductility of construction material are the fundamental guarantee for engineering reliability. With the development of civil engineering, the type and form of civil engineering materials are unceasingly rich and the performance is enhanced also, and the corresponding research and application on experimental techniques, theoretical analysis, numerical simulation, and actual engineering are improved constantly. However, there are still more phenomena to be interpreted and more research issues should be given attention because the engineering material has microcrack, heterogeneity, and evolution in micromechanics and shows elastoplasticity, discreteness, and randomness in the micromechanics. In addition, the mechanical behaviors of engineering materials are also related to the size, the loading rate, the external environment, and other factors. Therefore, the mechanical properties of different engineering materials have obvious differences; the demand about the experiment and numerical simulation for various materials are diverse. Researchers normally investigate the mechanic property of the specified material, and the study on the generalized characteristics for various construction materials is still not intense.

The uniaxial stress-strain curve indicates the simple mechanic property of the material subjected to pure compress or pure tension; representing the basic material constitutive relation and the uniaxial stress-strain relation is also the foundation to establish multiple compound constitutive models. In the elastoplasticity or nonlinear analysis, the uniaxial stress-strain curve plays a decisive role in the accuracy of the numerical results. Furthermore, the computational analysis of the engineering structure subjected to dynamic or cyclic loadings requires the stress-strain models to simulate the response of the structure [2, 3].

There are many types of stress-strain curves models for different material or even one kind of material. As known to all, the stress-strain curve of concrete has a variety of forms or functions because concrete is a type of composite material and the discreteness is obvious. Numerous concrete models have been proposed in the last years. For example, the compressive stress-strain function for concrete consists of five kinds of equations as polynomial, exponential, and trigonometric function, rational fraction, and sectional form and the total number of the corresponding formulas is over 20 [2].

In the macroscopic level, three broad categories can be distinguished: models derived from the theory of elasticity, models based on the theory of plasticity, and models based on the continuum damage theory [3, 4]. Also, some coupled models based on the association of plasticity and continuum damage theory have been recently developed. Although it has been proved that the models derived from theory of plasticity and continuum damage theory can accurately simulate the observed behavior of concrete, the engineering application of these models is less. This is motivated by the great amount of parameters that are usually needed and the difficulty to obtain them through conventional laboratory tests. From the perspective of another point of view, the structural member with definite shape and materials is a complex system which has huge amounts of units, and the units have the whole process of sustained force, damage, interact, transmitting energy, and propagation, until failure. The characteristics of all the units generally constitute the property of the macromaterial and member.

Hence, taking the damage propagation and transmit in the microunits into consideration, establishing a new and generalized uniaxial constitutive model which can take into account that both the versatility and the dirigibility is necessary.

2. SIR Model and Its Connection between Constitutive Relations of Material

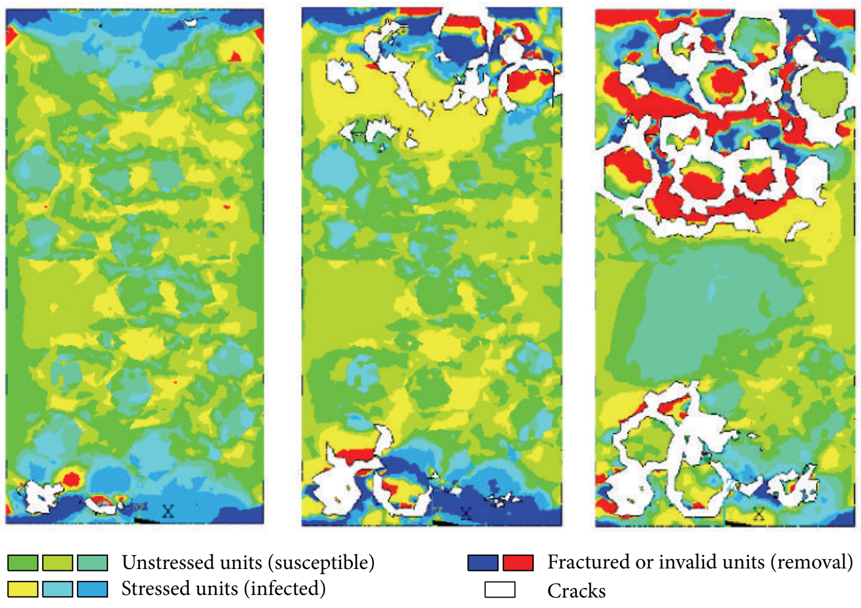
An epidemic represents the sudden outbreak and propagation of a disease, often occurring on a short temporal scare and affecting a significant portion of a population. Epidemics also may exhibit some periodic behavior, as opposed to endemics, which are diseases that are always present to some extent in a population. Epidemiology is the branch of science which essentially deals with the mathematical modeling of propagation of diseases. The first mathematical model of epidemiology was formulated and solved by Daniel Bernoulli in 1760. Kermack and McKendrick [5] illustrated that diseases showed a threshold type of behavior. In other words, if a single person infected by a particular disease passed on the infection to more than one person in turn, an epidemic would occur, while if less than one secondary infection occurred in each primary one, the disease would die out. The study of mathematical epidemiology has grown rapidly, with a large variety of models having been formulated and applied to infectious diseases. The epidemic model can describe the dynamic process of the epidemic, and spread characteristics will be grasped based on the information of the change of population; then the epidemic can be effectively controlled. The general epidemic model belongs to first order ordinary differential equations, and the most representative is the SIR model.

The SIR epidemic mathematical model is presented based on a system of first order ordinary differential equations and it has been used in the modeling of several infectious diseases, where the parameters need to be estimated by epidemiological data [6]. In this model, the variables represent subpopulations of the Susceptible (S) who can catch the disease, the Infected (I) who are infected and can transmit the disease to the Susceptible, and the Removal (R) who had the disease and recovered or died or have developed immunity or have been removed from contact with the other classes. Thus, the model describes the propagation and transformation between different classes. Assume that the total number of population is one unit for the sake of simplicity and the SIR model is written as

$$\begin{aligned} \frac{di(t)}{dt} &= \lambda i(t)s(t) - \mu i(t), \\ \frac{ds(t)}{dt} &= -\lambda i(t)s(t) + \eta i(t) - \rho i^2(t)s(t), \end{aligned} \tag{1}$$

where $i(t)$ and $s(t)$ denote the Infected and the Susceptible, respectively, and $r(t)$ is Removal and t is time. Hence, $s(t) + i(t) + r(t) = 1$ is satisfied. λ is the infectivity coefficient of the typical Lotka-Volterra interaction term, which means the daily contact rate (i.e., the population that each patient effectively contact with the healthy persons every day). μ is the daily recovery rate or number of patients cured or removed out accounting for the proportion of the total patients. η is the increasing rate of Susceptible persons when patients are increasing. ρ is the speed parameter after taking preventive and control measures. The SIR model is subject to the initial conditions $i(0) = i_0$ and $s(0) = s_0$, where $i_0 > 0$ and $s_0 > 0$ are given constants.

From the introduction above, it is significant that the SIR model is not simply an equation representing the spread of an epidemic, while it actually describes the propagation and dynamic evolution process of the different internal components and the characteristics in a complex system subject to external effect. In general, the SIR model embodies the general characteristics and rules in the similar system with propagation and transformation. The continuous damage and destruction will occur for the material specimen in civil engineering under increasing load, and the unstressed units, the stressed units, and the fractured or invalid units in anyone state should be included. The stress-strain relationship is the typical representation of the propagation and the dynamic evolution. Therefore, an innovative model for the material can be presented referencing to the SIR model, and the two models have similar parameters. To validate this interpretation, a compressed concrete specimen with 60 mm diameter and 120 mm height is selected as an example. The uniaxial failure process of the specimen is simulated based on the three-dimensional mesolevel finite element method [7, 8], and the failure charts in different stage are shown in Figure 1.



Unstressed units (susceptible)
Stressed units (infected)
Fractured or invalid units (removal)
Cracks

(a) Initial stage (b) Stressed stage (c) Failure stage

FIGURE 1: Damage propagation process and different units based on microlevel simulation.

It is clear that there are unstressed units, stressed units, and the fractured or invalid units in all stages, whereas the damage propagation and the proportion of each part are time-variant. The unstressed units are majority and fractured units are few at the initial stage. With the compression being enhanced, the units participate in the resistance gradually and the stressed units boom. After the counteragent in most units decreases, the invalid units increase and the cracks appear, though the stressed units decline and the specimen collapses. In conclusion, the SIR model is suitable for illustrating the failure process of materials, that is, typical damage propagation process in a complex system.

Strain ε reflects the deformation capability and damage process of material, corresponding with the time variable t in the SIR model. $s(\varepsilon)$, $\sigma(\varepsilon)$, and $r(\varepsilon)$ represent the equivalent stress in unstressed units, the equivalent stress in stressed units, and the equivalent stress in fractured or invalid units, respectively. This model subjects to the initial conditions $\sigma(0) = \sigma_0$ and $s(0) = s_0$, where $\sigma_0 > 0$ and $s_0 > 0$ are given constants. In the same manner, $s(\varepsilon) + \sigma(\varepsilon) + r(\varepsilon) = 1$. In this case, λ can be viewed as the transmissibility rate in units, and μ represents the failure rate in units. η can be expressed as the increase rate of stressed units when the invalid units grow, and ρ is the reduced speed parameter for invalid units due to stress transmission and distribution. Finally, the stress-strain model of material can be expressed as

$$\begin{aligned}\frac{d\sigma(\varepsilon)}{d\varepsilon} &= \lambda\sigma(\varepsilon)s(\varepsilon) - \mu\sigma(\varepsilon),\\ \frac{ds(\varepsilon)}{d\varepsilon} &= -\lambda\sigma(\varepsilon)s(\varepsilon) + \eta\sigma(\varepsilon) - \rho\sigma^2(\varepsilon)s(\varepsilon).\end{aligned} \tag{2}$$

Equation (2) is nonlinear differential equations which represent the coupling and transformation of the three types of units and the full complex process for the material from intact to failure. The system of equations has the unique solution, but the analytical solution cannot be obtained by normal mathematical methods and the results are calculated usually by numerical methods.

Assume the basic condition of the equation above is that λ is 1.00, μ is 0.15, η is 0.1, ρ is 0.8, and $\sigma(0)$ is 0.02. The consequences of the equivalent stress $\sigma(\varepsilon)$ under different parameters are solved by numerical method, as shown in Figure 2. It is obvious that the full curves accurately indicate the linear ascent stage, nonlinear yield stage, and the decline stage or strengthening stage after peak stress. Further, the variation of μ, η, or ρ can obviously affect peak and ductility of the stress-strain curve, and different forms of the curve can be obtained according to the overall adjustment of the corresponding parameters.

Different solutions of SIR model in various conditions can represent the stress-strain curves of diverse types of materials. Figure 3 reveals the stress-strain curves including concrete or rock-soil in compression or tension as well as different types of stress-strain curves of soils and steel obtained by numerical methods. The comparison curves of the real experimental data and numerical solution about the concrete and steel are shown in Figures 4 and 5, respectively. In order to compare the regularity but not the specific data, the peak strain is normalized. By changing the parameter appropriately, the full stress-strain curve obtained from (2) can simulate the peak stress, ductility, and softening stage of concrete with different strength, as shown in Figure 6. The results above show that the SIR model can really represent different mechanical properties of materials generally. It is worth noting that the initial values of the curves obtained by (2) are adjusted to the origin of coordinate and the coordinate values only have

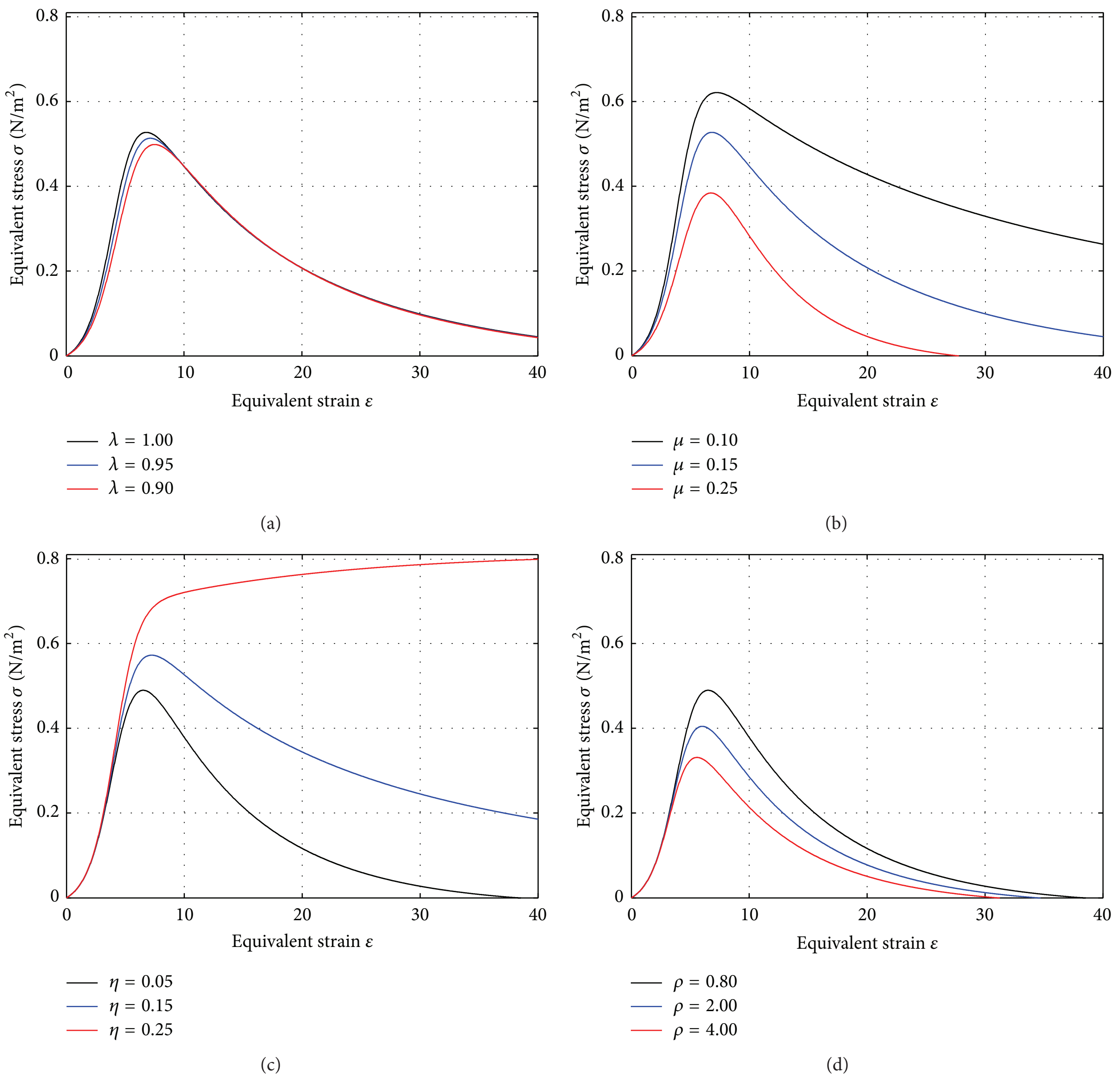


Figure 2: Numerical solutions of SIR for different parameter.

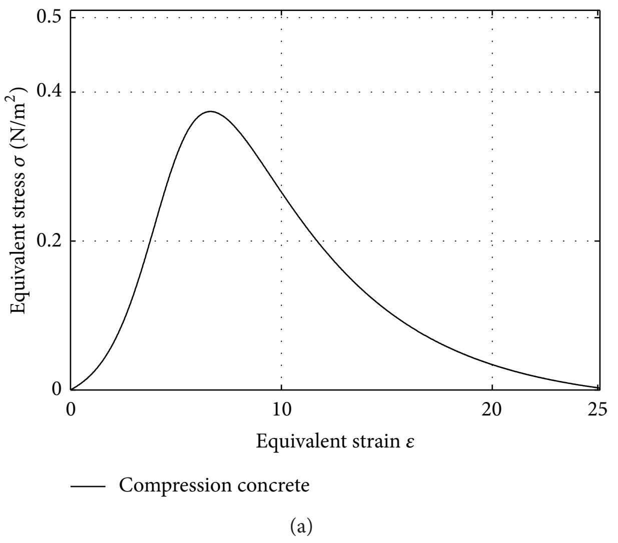
relative significance; it can be adjusted to a proportional coefficient to real physical value in practical applications.

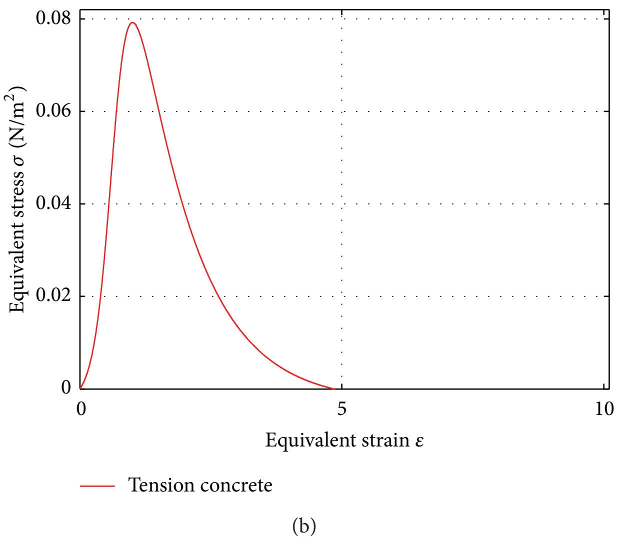
3. Analytical Solution and Approximate Solution of SIR Model

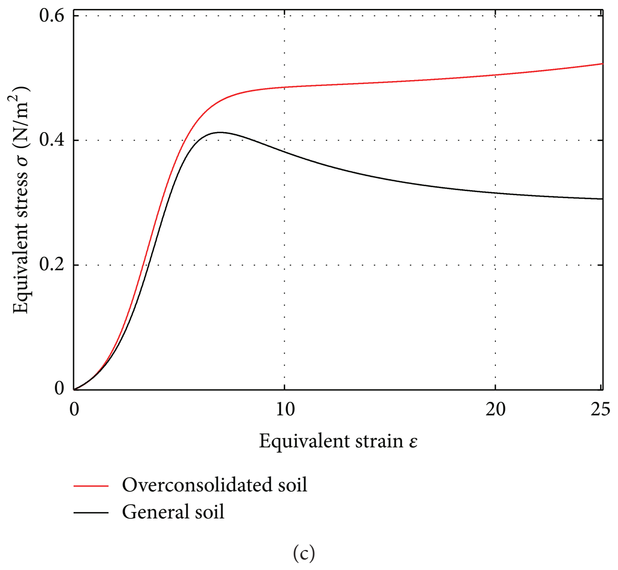
The numerical solution of the SIR model is accurate but it must be calculated for any specific case, not representing the panorama and rule of the solution. Therefore, the analytical solution of the SIR model is necessary and it has important mathematical and mechanical meaning. Many corresponding studies of the analytical solution of the SIR model have been carried out, but the problem is not solved perfectly because the model has strong nonlinearity. There are some classic nonlinearity techniques that can be considered, such as Adomian decomposition method [9, 10], delta expansion method, and perturbation method [11, 12]. All these methods have some limitations, such that these techniques do not provide a convenient way to adjust and control the convergence region and rate of approximation series.

To overcome the mentioned limitations, Liao has proposed the homotopy analysis method (HAM) for nonlinear problems and then modified it step by step [13], and many nonlinear problems in different fields have been successfully solved by HAM.

Many problems such as boundary layer similarity solution for forced, natural, and mix convection in porous medium, heat transfer, and fluid mechanic problems are nonlinear inherently. The analytical solutions of nonlinear







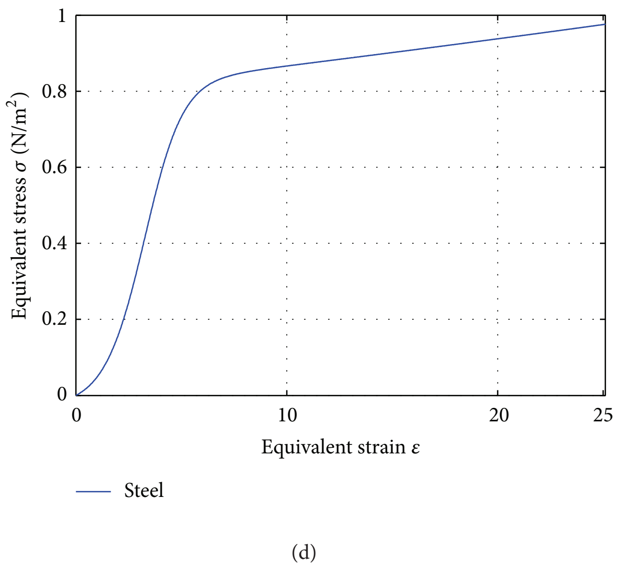


Figure 3: Constitutive curves for different engineering material based on SIR model.

ordinary differential equations can be solved by HAM [14–16].

HAM is based on homotopy, a concept in topology. The notion of equivalent maps or processes, where one can be deformed into the other, is the fundamental structure of homotopy. Two maps $f, g : X \rightarrow Y$ of topological spaces are homotopic if there exists a map $\phi : X \times I \rightarrow Y$ such that $\phi(x, 0) = f(x)$ and $\phi(x, 1) = g(x)$ for $x \in X$. Here, $X \times I$ denotes the product of X with unit interval $[0, 1]$ of real numbers. The map ϕ is called the homotopy between f and g. The basic idea of HAM is shown in the nonlinear differential equation as follows:

$$N[f(\vec{r}, t)] = 0, \quad (3)$$

where N is nonlinear operator, $\vec{r}$ is a vector of spatial variables, t denotes time, and $f(\vec{r}, t)$ is an unknown function. Boundary or initial conditions can be treated in a similar manner, which we avoid here just for simplicity [17].

Generalizing the concept of traditional homotopy, the so-called zero-order deformation equation is established as

$$(1 - q) L[\phi(\vec{r}, t; q) - f_0(t)] = q\hbar H(\vec{r}, t) N[\phi(\vec{r}, t; q)], \quad (4)$$

where $q \in [0, 1]$ is the embedding parameter, $\phi(\vec{r}, t; q)$ is an unknown function, $H(\vec{r}, t)$ is an auxiliary function, L is a linear operator, $f_0(t)$ is the initial guess, and $\hbar$ is a convergence-control parameter. As q increases from 0 to 1, $\phi(\vec{r}, t; q)$ vary from initial trial $f_0(t)$ to the exact solution $f(\vec{r}, t)$. If this variation is smooth enough, we construct the Maclaurin series of $\phi(\vec{r}, t; q)$ at $q = 0$ and the coefficients of all the higher terms can be obtained from the higher-order deformation equations. Differentiating the zeroth-order deformation equation above m times with respect to the embedding parameter q, then setting $q = 0$, and finally

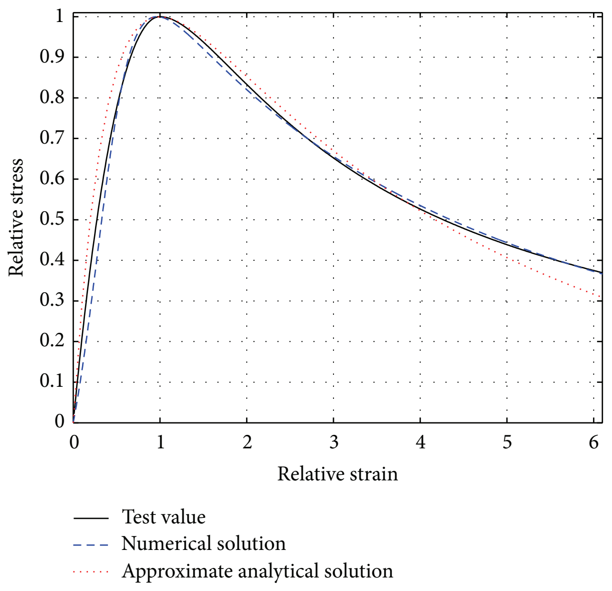


FIGURE 4: Concrete constitutive curves for different results.

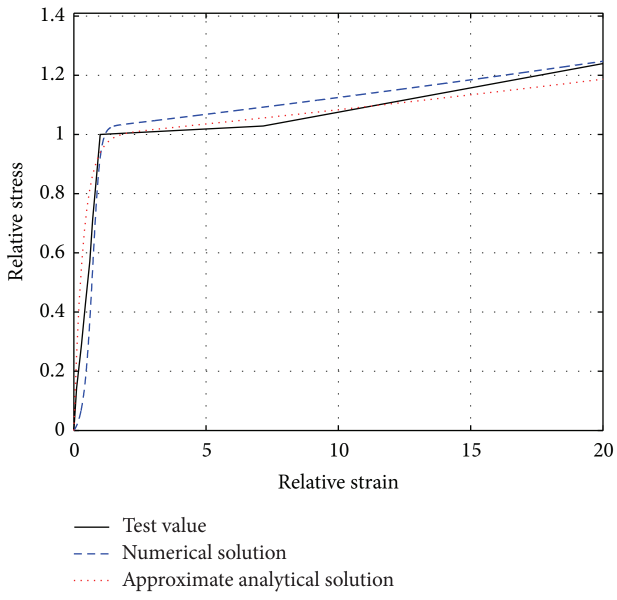


FIGURE 5: Steel constitutive curves for different results.

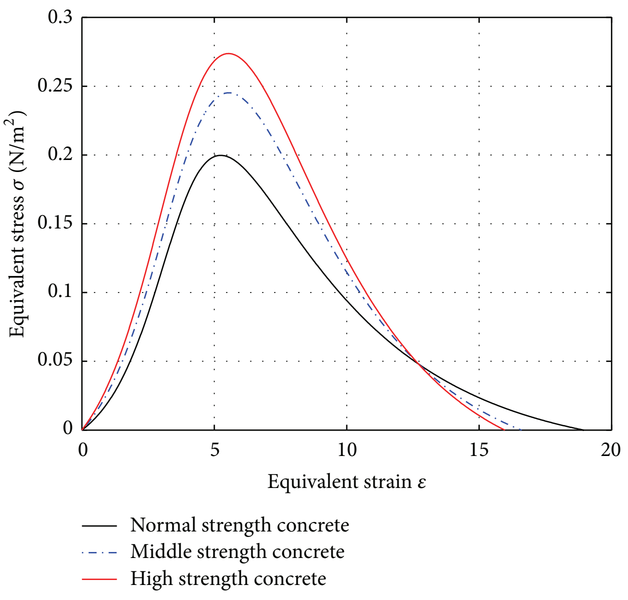


FIGURE 6: Constitutive curves for concrete based on SIR model.

dividing by $m!$, we have the so-called mth-order deformation equations:

$$L\left[\phi_m(\vec{r},t;q)-\chi_m\phi_{m-1}(\vec{r},t;q)\right]=\hbar H(\vec{r},t)\,R_m \qquad (5)$$

with $R_m=(\partial^m N[\phi(\vec{r},t;q)]/(m!\partial q^m))|_{q=0}=m!\cdot f_m(t)$.

In this way, a nonlinear equation is transformed into a series of linear equations. The exact solution of $f(\vec{r},t)$ is then approximated by the summation of the Maclaurin series at $q=1$. One has great freedom for the choice of linear operator L. For example, one can even select linear operator of different order as compared to the original nonlinear problem. To simplify the applications of HAM, Liao has suggested some rules, that is, the rule of solution expression, the rule of solution existence, and the rule of ergodicity for coefficients of homotopy series solution [13].

In HAM, there is sufficient space for the convergence of the approximation by the introduction of $\hbar$ which greatly improves the early homotopy analysis method. It provides us with a simple way to ensure the convergence of the series solutions of nonlinear problems. This is an obvious advantage of HAM over homotopy perturbation method (HPM). In fact, HPM is just a special case of HAM when $\hbar=-1$, as pointed out by Liao and Abbasbandy and, in general, proved by Sajid and Hayat [18, 19]. Liao also has pointed out that HAM logically contains other nonperturbation methods such as Adomian's decomposition method [9], the δ-expansion method, and Lyapunov's artificial small parameter method [12].

From (2), we are led to define the two nonlinear operators as

$$\begin{aligned}
&N_S\left[\sigma(\varepsilon;q),S(\varepsilon;q)\right]\\
&\quad=\frac{\partial\sigma(\varepsilon;q)}{\partial\varepsilon}-\lambda\sigma(\varepsilon;q)\,S(\varepsilon;q)+\mu\sigma(\varepsilon;q),\\
&N_I\left[\sigma(\varepsilon;q),S(\varepsilon;q)\right]\\
&\quad=\frac{\partial S(\varepsilon;q)}{\partial\varepsilon}+\lambda\sigma(\varepsilon;q)\,S(\varepsilon;q)-\eta\sigma(\varepsilon;q)\\
&\qquad+\rho\sigma^2(\varepsilon;q)\,S(\varepsilon;q).
\end{aligned} \qquad (6)$$

Let $s_0(\varepsilon)$ and $\sigma_0(\varepsilon)$ denote the initial guesses of $s(\varepsilon)$ and $\sigma(\varepsilon)$, L_S and L_σ the two auxiliary linear operators, $H_S(\varepsilon)$ and $H_\sigma(\varepsilon)$ the two auxiliary functions, and $\hbar$ an auxiliary parameter, all of which will be determined later. Let $q\in[0,1]$;

denote the embedding parameter. We construct the wroth-order deformation equations:

$$(1-q) L_S [S(\varepsilon; q) - s_0(\varepsilon)] = q\hbar H_S(\varepsilon) N_S [S(\varepsilon; q), \sigma(\varepsilon; q)], \tag{7}$$

$$(1-q) L_\sigma [\sigma(\varepsilon; q) - \sigma_0(\varepsilon)] = q\hbar H_\sigma(\varepsilon) N_\sigma [S(\varepsilon; q), \sigma(\varepsilon; q)], \tag{8}$$

subject to the initial conditions $S(0; q) = s(0)$ and $\sigma(0; q) = \sigma(0)$.

Expand $S(\varepsilon; q)$ and $\sigma(\varepsilon; q)$ in the Taylor series with respect to q; assuming that $H_S(\varepsilon)$ and $H_\sigma(\varepsilon)$ are properly chosen so that the above two series converge at $q = 1$, the solution series are

$$S(\varepsilon) = s_0(\varepsilon) + \sum_{m=1}^{+\infty} s_m(\varepsilon),$$
$$\sigma(\varepsilon) = \sigma_0(\varepsilon) + \sum_{m=1}^{+\infty} \sigma_m(\varepsilon). \tag{9}$$

Differentiating the wroth-order deformation equations (7) and (8) m times with respect to the embedding parameter q, then setting $q = 0$, and finally dividing by $m!$, the so-called mth-order deformation equations are

$$L_S [s_m(\varepsilon) - \chi_m s_{m-1}(\varepsilon)] = \hbar H_S(\varepsilon) R_m^S(\varepsilon), \tag{10}$$

$$L_\sigma [\sigma_m(\varepsilon) - \chi_m \sigma_{m-1}(\varepsilon)] = \hbar H_\sigma(\varepsilon) R_m^\sigma(\varepsilon), \tag{11}$$

subject to the initial conditions $\sigma_m(0) = 0$ and $s_m(0) = 0$, where $R_m^S(\varepsilon) = s'_{m-1}(\varepsilon) + \lambda \sum_{k=0} \sigma_k(\varepsilon) s_{m-1-k}(\varepsilon)$ and $R_m^\sigma(\varepsilon) = \sigma'_{m-1}(\varepsilon) + \mu\sigma_{m-1}(\varepsilon) - \lambda \sum_{k=0} \sigma_k(\varepsilon) s_{m-1-k}(\varepsilon)$. χ_m is 0 when m is not greater than 1, and χ_m is 1 in other cases.

Thus, $s(\varepsilon)$ and $\sigma(\varepsilon)$ can be expressed as

$$s(\varepsilon) = s(\infty) + \sum_{k=1}^{+\infty} b_k e^{-k\beta\varepsilon},$$
$$\sigma(\varepsilon) = \sum_{k=1}^{+\infty} a_k e^{-k\beta\varepsilon}. \tag{12}$$

From (1) and (2), $\sigma'(0) = \lambda\sigma(0)S(0) - \mu S(0)$ and $S'(0) = -\lambda\sigma(0)S(0) + \eta\sigma(0) - \rho\sigma^2(0)S(0)$. To obtain solutions in the form of (12), the initial guesses $s_0(\varepsilon)$ and $\sigma_0(\varepsilon)$ are expressed as follows:

$$s_0(\varepsilon) = s(\infty) + \gamma_{0,1} e^{-\beta\varepsilon} + \gamma_{0,2} e^{-2\beta\varepsilon}, \tag{13}$$

where $\gamma_{0,1} = 2(s(0) - s(\infty)) - (\lambda s(0) + \rho s(0) i(0) - \eta) i(0)/\beta$ and $\gamma_{0,2} = s(\infty) - s(0) + (\lambda s(0) + \rho s(0) i(0) - \eta) i(0)/\beta$.

Consider

$$\sigma_0(\varepsilon) = \delta_{0,1} e^{-\beta\varepsilon} + \delta_{0,2} e^{-2\beta\varepsilon}, \tag{14}$$

where $\delta_{0,1} = 2\sigma(0) + \sigma(0)(\lambda s(0) - \mu)/\beta$ and $\delta_{0,2} = -\sigma(0) - \sigma(0)(\lambda s(0) - \mu)/\beta$.

By choosing the auxiliary linear operators and from deformation equations and deformation derivative condition, we have

$$s_m(\varepsilon) = \sum_{k=1}^{3m+2} \gamma_{m,k} e^{-k\beta\varepsilon},$$
$$\sigma_m(\varepsilon) = \sum_{k=1}^{3m+2} \delta_{m,k} e^{-k\beta\varepsilon}, \tag{15}$$

where $\delta_{m,k}$ and $\gamma_{m,k}$ are coefficients. Substituting the above expressions into (10) to (11), the recurrence formulas are

$$\delta_{m,1} = -\sum_{j=2}^{3m+2} \delta_{m,j},$$
$$\gamma_{m,1} = -\sum_{j=2}^{3m+2} \gamma_{m,j},$$
$$\delta_{m,j} = \chi_m \chi_{3m-j+1} \delta_{m-1,j} - \left(\frac{\hbar}{\beta}\right) \frac{a_{m,j-1}}{j-1}, \quad 2 \le j < 3m+2,$$
$$\gamma_{m,j} = \chi_m \chi_{3m-j+1} \gamma_{m-1,j} - \left(\frac{\hbar}{\beta}\right) \frac{b_{m,j-1}}{j-1}, \quad 2 \le j < 3m+2, \tag{16}$$

where a and b are parameters to meet deformation equation.

Finally, we get analytic solution of expressions in (2):

$$\sigma(\varepsilon) = \sum_{m=1}^{+\infty} \sum_{k=1}^{3m+2} \delta_{m,k} e^{-k\beta\varepsilon},$$
$$s(\varepsilon) = s(0) + \sum_{m=1}^{+\infty} \sum_{k=1}^{3m+2} \gamma_{m,k} e^{-k\beta\varepsilon}. \tag{17}$$

At this point, $\beta = \mu - \lambda s(\infty) \approx \mu$. According to the above conclusions and characteristics of solution, the approximate analytical solution of (17) can be expressed as

$$\sigma(\varepsilon) = \sum_{i=1}^{n} (-1)^{i+1} c_i e^{-k_i \mu\varepsilon}, \tag{18}$$

where n belongs to even number and $c_i = c_{i+1}$ when i is an odd number. The above approximate analytical solution can usually achieve adequate satisfactory results when n is 2 or 4. As a simplified format, the equation above can also be transformed as follows:

$$\sigma(\varepsilon) = c_1 \left(e^{-k_1 \mu\varepsilon} - e^{-k_2 \mu\varepsilon}\right). \tag{19}$$

The comparison of the real concrete strain-stress curve, the numerical solution curve, and the approximate analytical solution of the SIR model is shown as Figure 4, and the comparison of steel is shown in Figure 5. It can be seen that both the numerical solution and the approximate analytical solution fit the original value precisely. It is worth noting that the approximate analytical solution reflects the

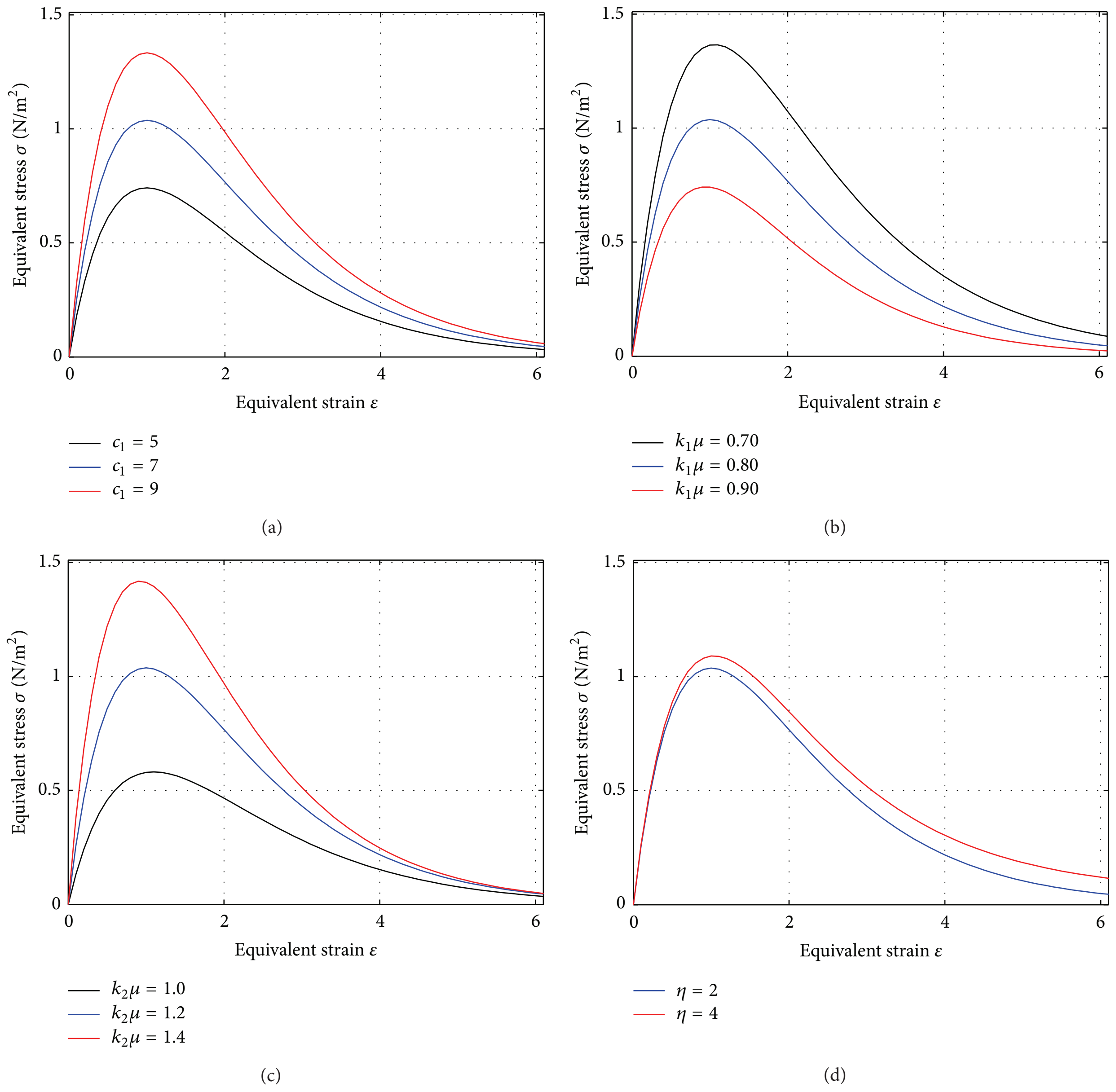


FIGURE 7: Approximate solutions based on SIR model with different parameter.

connotation and characteristics of the analytical solutions, but it nevertheless has a certain similarity, and the specific value of c_i in the solution should be adjusted appropriately according to actual condition and the numerical solution in order to obtain better accuracy. In addition, Umemura and Aoyama [20] have presented an exponential constitutive relation for concrete, which is similar with the model as (18) in n equals 2. However, the original exponential relation is determined by experimental data fitting, and the proposed solution in this paper has the theoretical basis and generality; the results can be verified with each other. The effect of various parameters in (18) on solutions is shown in Figure 7. The stress-strain curve of compression concrete with different strength is shown as Figure 8, and the simulation curves are similar with the experimental data in shapes and the variation rules, indicating that the approximate analytical solution put forward in this paper can also embody the intrinsic characteristics and variation of the SIR model.

4. Size Effect and Strain Rate Based on SIR Solution

For real engineering materials, the mechanical properties also related to size, load mode, and external environment besides their own composition and characteristics and mainly include size effects, strain rate, and multiaxis loading. In this paper, the uniaxial stress-strain curve on the influence of size effect and strain rate of are mainly discussed.

The size effect of material strength refers to phenomenon that large size member in brittle material usually fractures

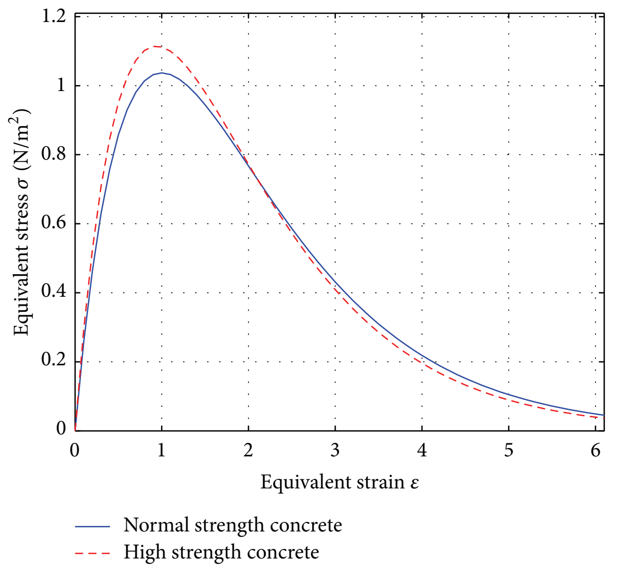


Figure 8: Constitutive relation of concrete with different strength based on approximate solutions.

under a lower nominal stress than geometrically similar small-size member (the nominal stress being defined as the load divided by the characteristic cross section area). Early researchers suppose that any observed size effect should be described by extreme value statistics prevailed in structural engineering. In the mid 1970s, the fact that there exists a purely deterministic size effect, caused by energy release associated with stress redistribution prior to failure and that this energetic size effect usually dominates in the so-called quasibrittle structures (i.e., structures in which fracture propagation is preceded by a relatively large fracture process zone which, in contrast to brittle ductile fracture of metals, exhibits almost no plastic deformations but undergoes progressive softening due to microcracks) gradually emerged.

Bažant and Chen [21] and Bažant and Planas [22] summarized six main causations for size effect which include boundary layer, diffusion phenomena, hydration heat, randomness of material strength, energy release, and the fractal character of the crack surface. Recent research focuses on three main types of size effects, namely, the statistical size effect due to randomness of strength, the energy release size effect, and the possible size effect due to fractality of fracture or microcracks.

Recent research has shown that the elastic modulus and the peak stress of brittle material will gradually decrease with the increase of the member size and depth-width ratio; at the same time, the strain value at the peak stress changes a little, and the descent rate of stress and fragility at the softening section will also reduce [23], as shown in Figures 2 and 7. In some cases, the curves of different sizes in softening section can even intersect as shown in Figures 6 and 8.

In fact, for the large size member with the same axial force, the total number and size of microcracks are bigger and the domain that the sustained damage occurs is larger. Hence, the equivalent strain energy for damage needs fewer paths and the damage propagation needs shorter paths, which lead to the probability of regional brittle failure; that is, the failure rate in unit and its growth rate both increase.

Considering the above characteristic of size effect, the SIR model can simulate the size effect of brittle material and the whole process of failure. The advantage of the application of the SIR model is the physical significance being explicit and the variation can be realized by enhancing the failure rate in unit μ and the reduced speed parameter for invalid elements due to stress transmission and distribution ρ, referring to Figure 2.

The materials such as concrete and rocks are typical rate sensitive for their strength, ductility, and failure mode will significantly change in different strain rates. Existing research shows that the elastic modulus and the ultimate strength of rate sensitive material will enhance with increasing strain rate, and the strain value at the peak stress changes a little [23]. In different strain rates, the stress-strain full curve of concrete is basically consistent with the whole curve under static load in shape; however, the ductility increases slightly, as shown in Figures 2 and 7.

For the materials subjected to the same axial force, if the strain rate increased, the internal microcracks in the mortar substrate are late to fully extent, but the quantity and degree of the damaged aggregate relatively increase which lead to the enhancement of the failure strength. At the same time, the unit failure rate increased, and the increase rate of the stressed unit sustainably grows.

Therefore, the SIR model can show the influence of the strain rate from the physical sense by properly adjusting parameters, and the simulation is realized by enhancing the failure rate μ and the increase rate of stressed units when the invalid units grow η, referring to Figure 2.

The above discussion illustrates the direct relationship between size effect, strain rate, and the SIR model with its numerical solution from mechanical principle and propagation characteristic. In practical applications, the approximate analytic solution of the model is more convenient, so the study below will discuss the adjustment of coefficient c_i in the approximate analytic solution for the purpose of representing size effect and strain rate.

The most widely used theory is the size effect law proposed by Bažant and Chen [21] based on a large number of experiments, and this method put forward the size effect unified formula in certain extent according to plasticity theory or elastic theory:

$$\frac{\sigma_N}{f_c} = \frac{B}{\sqrt{1 + (D/D_0)}}, \tag{20}$$

where σ_N is the nominal stress when material is damaged, B is dimensionless parameter, f_c is the strength of the quasi brittle material, D is the characteristic length of the structure, and D_0 is the constant related to structural shape. It can be seen from Figure 6 that the consideration about size effect can be realized by modulating c_i and k_i properly in the approximate analytical solution of SIR.

In the usual study, the strain rate effect coefficient of compression concrete is described as the following exponential form according to European standard [24]:

$$\frac{\sigma_{\mathrm{cd}}}{\sigma_{\mathrm{cs}}} = \left(\frac{\dot{\varepsilon}}{\dot{\varepsilon}_0}\right)^{1.026\alpha}, \quad \dot{\varepsilon} \le 30/\mathrm{s},$$
$$\frac{\sigma_{\mathrm{cd}}}{\sigma_{\mathrm{cs}}} = \gamma_s\left(\frac{\dot{\varepsilon}}{\dot{\varepsilon}_0}\right)^{1/3}, \quad \dot{\varepsilon} > 30/\mathrm{s}, \tag{21}$$

where $\dot{\varepsilon}$ is the current material strain rate, $\dot{\varepsilon}_0$ is the quasi static strain rate, taken as 3×10^{-5}/s, and σ_{cs} and σ_{cd} are the static and dynamic compression strength of concrete, respectively. One has $\alpha = 1/(5 + 0.9\,\sigma_{\mathrm{cs}})$ and $\log\gamma_s = 6.156\,\alpha - 2$.

For tensioned concrete

$$\frac{\sigma_{\mathrm{td}}}{\sigma_{\mathrm{ts}}} = \left(\frac{\dot{\varepsilon}}{\dot{\varepsilon}_0}\right)^{1.016\delta}, \quad \dot{\varepsilon} \le 30/\mathrm{s},$$
$$\frac{\sigma_{\mathrm{td}}}{\sigma_{\mathrm{ts}}} = \beta\left(\frac{\dot{\varepsilon}}{\dot{\varepsilon}_0}\right)^{1/3}, \quad \dot{\varepsilon} > 30/\mathrm{s}, \tag{22}$$

where $\dot{\varepsilon}$ is the current material strain rate and $\dot{\varepsilon}_0$ is the quasi static strain rate, taken as 3×10^{-5}/s, and σ_{cs} and σ_{cd} are the static and dynamic tension strength of concrete, respectively. One has $\delta = 1/(10 + 0.6)\,\sigma_{\mathrm{ts}}$ and $\log\beta = 7.11\delta - 2.33$.

The strain rate effect coefficient of steel I is as follows:

$$\frac{f_{\mathrm{yd}}}{f_{\mathrm{ys}}} = \left(1 + \frac{d_1}{f_{\mathrm{ys}}\ln(\dot{\varepsilon}/\dot{\varepsilon}_0)}\right)$$
$$\frac{f_{\mathrm{ud}}}{f_{\mathrm{us}}} = \left(1 + \frac{d_2}{f_{\mathrm{us}}\ln(\dot{\varepsilon}/\dot{\varepsilon}_0)}\right), \tag{23}$$

where $\dot{\varepsilon}$ is the current material strain rate and $\dot{\varepsilon}_0$ is the quasi static strain rate, taken as 3×10^{-4}/s, and f_{ys} and f_{yd} are the static and dynamic strength, respectively. d_1 and d_2 are the test parameters obtained by regression analysis method [25, 26].

Referring to Figure 7, the strain rate effects on concrete can be simulated by modulating c_i and k_i properly for the approximate analytical solution of SIR model. In addition, if k_i is taken as a small negative number when i is an odd, the characteristics of strengthening after yielding for metal materials can also be simulated, and the stress-strain curve of metal with high strain rate or high strength can be obtained by enlarging the absolute value of c_i or k_i, as shown in Figure 9.

In conclusion, it is significant that the coefficient c_i of the approximate analytical solution implies the multiple parameters interaction in SIR model, and the size effect and strain rate in the whole load process can be achieved by regulating c_i. The factor ζ_s on size effect can be achieved from (19) and the factor ζ_d on strain rate can be achieved from (21) to (23). Thus, the correction coefficient about c_i is

$$c_{ic} = (k_s\zeta_s + k_d\zeta_d)\,c_i, \tag{24}$$

where k_s and k_d are adjustment coefficient on size effect and strain rate, respectively, and the more precise value should be determined by experimental data.

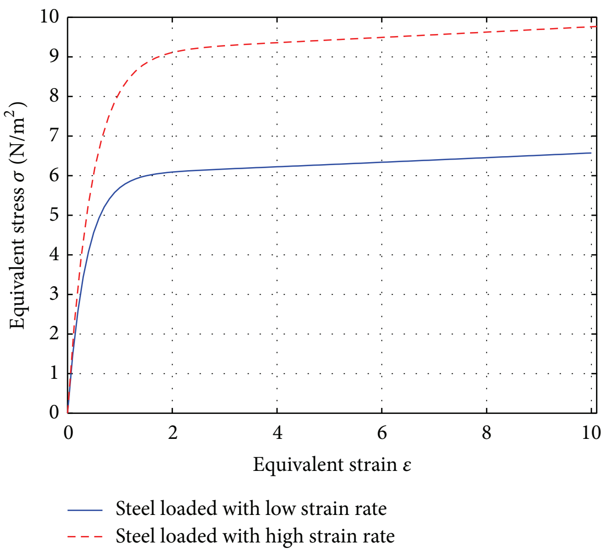


Figure 9: Constitutive relation of steels with different strain rate based on approximate solutions.

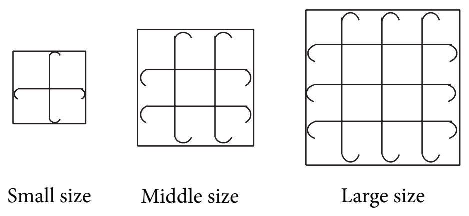


Figure 10: Configurations of stirrups.

5. Test on the Size Effect of Confined Concrete

As mentioned above, the uniaxial construction relationship of single material based on SIR model and HAM method is presented. However, the normal member of civil engineering consists of both concrete and steel, and the constructive relationship of reinforced concrete and confined concrete and corresponding properties such as size effect need more research.

In order to verify the feasibility of the construction relationship for confined concrete based on the SIR model, six reinforced concrete prism specimens confined by square stirrups were made. Each type of specimen contains two same members, and the volumetric percentage of stirrups is 1.26% for all. The configurations of stirrups are shown in Figure 10. The design parameters of the specimens are listed in Table 1. For the concrete used, the normal prismatic compressive strength is 42.67 N/mm^2, the ultimate compression strain is 0.0022, and Young's modulus is 3.08×10^4 N/mm^2. For the steel bars, the average yield strength is 480 N/mm^2, the ultimate strength is 665 N/mm^2, and Young's modulus is 2.05×10^5 N/mm^2.

The electrohydraulic servo testing machine with 4×10^4 kN maximum range was used as the load device and the continuous axial monotonic load was applied, as shown in Figure 11. When the force is less than the ultimate bearing

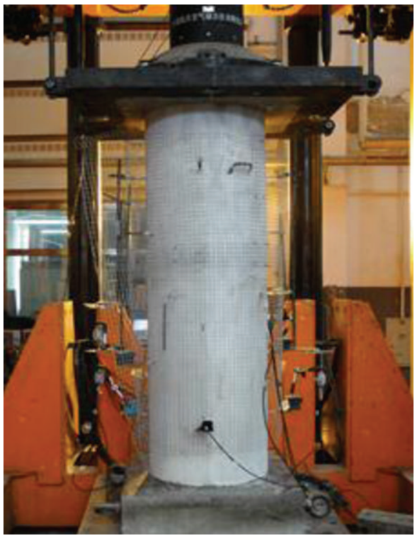


FIGURE 11: Loading and measuring instruments.

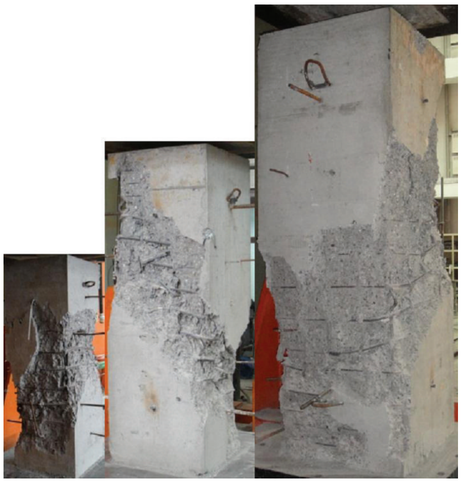


FIGURE 12: Final failure mode of specimens.

TABLE 1: Design parameters of the specimens.

Parameters (unit)	Notation	Specimen size		
		Small	Middle	Large
B (mm)	Section length	400	600	800
H (mm)	Height of specimen	1200	1800	2400
d_s (mm)	Diameter of stirrups	10	12	14
s (mm)	Spacing of stirrups	103	132	169
c (mm)	Thickness of protective layer	20	30	40
d_l (mm)	Diameter of longitudinal bars	12	18	22

capacity, the load was controlled by force and then by displacement. The measured parameters include axial loads, axial deformation, and stirrups strain. The final failure modes of specimens are shown in Figure 12.

The axial load value is measured by the strain force transducer with 1×10^4 kN measuring range. The axial

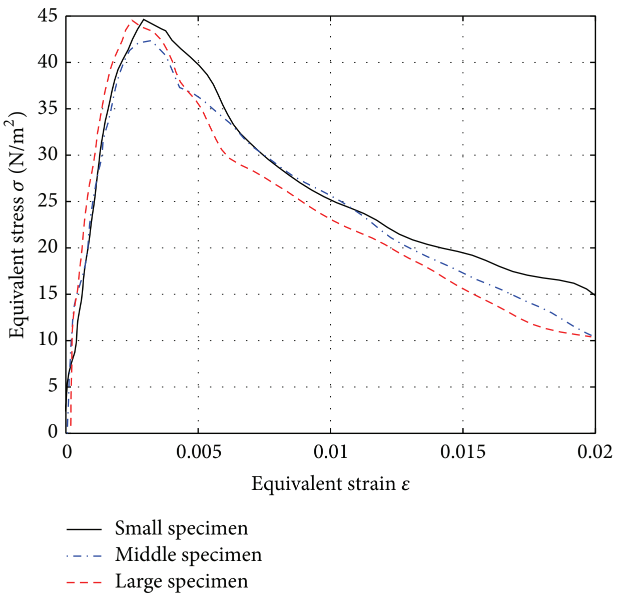


FIGURE 13: Stress-strain curve of gross area for all prism specimens.

TABLE 2: Parameters of the approximate solution.

c_1	k_1	k_2
1.345	2.305	20.546
1.386	2.727	23.785
1.391	3.119	28.267

compression is measured by the displacement meter with 200 mm measuring range and the gauge length is 2/3 of the specimen height. The average stress-strain relation of cross section curves of the specimens with different sizes is shown in Figure 13. The abscissa ε is obtained by the measured displacement divided by the respective gauge length. Ordinate σ is the relative stress, equal to the axial load value divided by the cross-sectional area of the specimen.

According to the stress-strain relation of the specimens with different sizes, the approximate solution based on nonlinear data fitting and (19) is presented as shown in Figure 14. The failure rate μ is assumed as 0.1, and the parameters in (19) are listed in Table 2. It is significant that the curves of the approximate solution fit well with the original data and the variation trend and the variance rule among the curves are clearly revealed. The SIR model and the corresponding approximate solution are suitable for confined concrete and complex composite material with size effect.

6. Conclusion

Though there are various constitutive relation models of engineering material at present, the united constitutive relation model is rare and it is necessary to establish a model which can present many properties in different materials by concise form.

Under the external load, the member composed of engineering materials is damaged inevitably and the damage

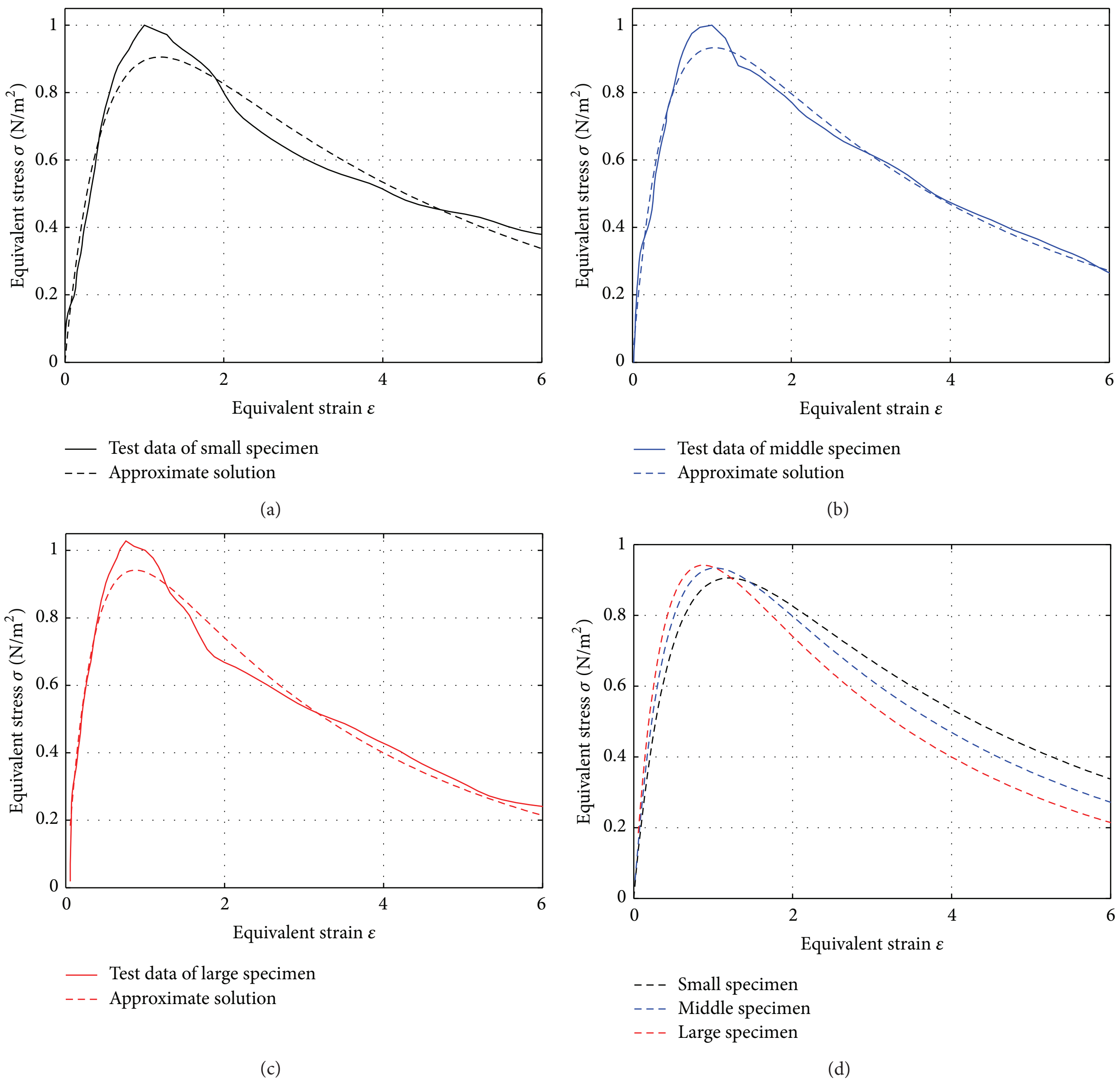


FIGURE 14: Stress-strain curve comparison of test value and approximate solution.

will propagate in the units and the system. This phenomenon has similar rule as the infectious disease. During infectious disease, the population includes the Susceptible people, the Infected people, and the Removal people. The SIR model originates from the infectious disease transmission dynamics and the complex system of population, which reflects the dynamic characteristics and the transmit laws of different parts of the system under the action of external factors and patterns. Hence, the SIR model can be used for reference and to indicate the full stress-strain curve in uniaxial material with high precision. The numerical results show that the constitutive relation in different materials can be simulated by choosing appropriate parameters in the SIR equations.

In this paper, the SIR models are described by coupled nonlinear differential equations, and the analytic solution form and approximate analytic solution of SIR model are obtained by means of an analytic technique for nonlinear problems, namely, the homotopy analysis method (HAM). According to the different solutions on the variation of the parameters in the SIR model, the mechanical characteristics of various materials are compared and the factors of the size effect and the strain rate are discussed. The results show that the SIR model and its solution presented in this paper have versatility and can provide unified constitutive relations for a variety of engineering materials.

However, the SIR model only has an implicit form, which differs with the traditional constitutive relations established by experimental results, damage mechanics, and fracture mechanics, and the intension and laws need further study. Furthermore, the SIR model and its solution are merely suitable for the materials under uniaxial load. The models for materials under multiaxial load and fatigue load are necessary to be developed and discussed.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

This work is partially supported by the Natural Science Foundation of China under Grant nos. 51108009 and 50878010, the Foundation of the Ministry of Education of China for Outstanding Young Teachers in University under Grant no. 20111103120022, and the Foundation of Beijing Key Lab of Earthquake Engineering and Structural Retrofit under Grant no. 2013TS02.

References

[1] R. K. Rajput, *Engineering Materials*, Chand and Company Ltd., New Delhi, India, 2008.

[2] W. F. Chen, *Plasticity in Reinforced Concrete*, J. Ross Publishing, Fort Lauderdale, Fla, USA, 2007.

[3] F. Lene, "Damage constitutive relations for composite materials," *Engineering Fracture Mechanics*, vol. 25, no. 5-6, pp. 713–728, 1986.

[4] N. Burlion, F. Gatuingt, G. Pijaudier-Cabot, and L. Daudeville, "Compaction and tensile damage in concrete: constitutive modelling and application to dynamics," *Computer Methods in Applied Mechanics and Engineering*, vol. 183, no. 3-4, pp. 291–308, 2000.

[5] W. O. Kermack and A. G. McKendrick, "Contribution to the mathematical theory of epidemics," *Proceedings of the Royal Society of London A*, vol. 115, pp. 700–721, 1927.

[6] H. W. Hethcote, "The mathematics of infectious diseases," *SIAM Review*, vol. 42, no. 4, pp. 599–653, 2000.

[7] Z. M. Wang, A. K. H. Kwan, and H. C. Chan, "Mesoscopic study of concrete I: generation of random aggregate structure and finite element mesh," *Computers and Structures*, vol. 70, no. 5, pp. 533–544, 1999.

[8] P. Wriggers and S. O. Moftah, "Mesoscale models for concrete: homogenisation and damage behaviour," *Finite Elements in Analysis and Design*, vol. 42, no. 7, pp. 623–636, 2006.

[9] G. Adomian, "Nonlinear stochastic differential equations," *Journal of Mathematical Analysis and Applications*, vol. 55, no. 2, pp. 441–452, 1976.

[10] G. Adomian, *Solving Frontier Problems of Physics: The Decomposition Method*, Kluwer Academic Publishers, Boston, Mass, USA, 1994.

[11] J. D. Cole, *Perturbation Methods in Applied Mathematics*, Blaisdell Publication Company, Waltham, Mass, USA, 1968.

[12] A. H. Nayfeh, *Perturbation Methods*, Wiley-Interscience, New York, NY, USA, 2000.

[13] S. Liao, *Beyond Perturbation: Introduction to the Homotopy Analysis Method*, Chapman & Hall, Boca Raton, Fla, USA, 2003.

[14] A. R. Sohouli, D. Domairry, M. Famouri, and A. Mohsenzadeh, "Analytical solution of natural convection of Darcian fluid about a vertical full cone embedded in porous media prescribed wall temperature by means of HAM," *International Communications in Heat and Mass Transfer*, vol. 35, no. 10, pp. 1380–1384, 2008.

[15] Z. Ziabakhsh and G. Domairry, "Solution of the laminar viscous flow in a semi-porous channel in the presence of a uniform magnetic field by using the homotopy analysis method," *Communications in Nonlinear Science and Numerical Simulation*, vol. 14, no. 4, pp. 1284–1294, 2009.

[16] A. Kargar and M. Akbarzade, "Analytic solution of natural convection flow of a non-newtonian fluid between two vertical flat plates using homotopy perturbation method (HPM)," *World Applied Sciences Journal*, vol. 20, no. 11, pp. 1459–1465, 2012.

[17] S. Liao and Y. Tan, "A general approach to obtain series solutions of nonlinear differential equations," *Studies in Applied Mathematics*, vol. 119, no. 4, pp. 297–354, 2007.

[18] S. Abbasbandy, "The application of homotopy analysis method to nonlinear equations arising in heat transfer," *Physics Letters A*, vol. 360, no. 1, pp. 109–113, 2006.

[19] M. Sajid and T. Hayat, "Comparison of HAM and HPM methods in nonlinear heat conduction and convection equations," *Nonlinear Analysis. Real World Applications*, vol. 9, no. 5, pp. 2296–2301, 2008.

[20] H. Umemura and H. M. T. Aoyama, *Experimental Studies on Reinforced Concrete Members and Composite Steel and Reinforced Concrete Members*, University of Tokyo Publication, Tokyo, Japan, 1970.

[21] Z. P. Bažant and E. P. Chen, "Scaling of structural failure," *ASME Applied Mechanics Reviews*, vol. 50, pp. 593–627, 1997.

[22] Z. P. Bažant and J. Planas, *Fracture and Size Effect in Concrete and Other Quasibrittle Materials*, CRC Press, Boca Raton, Fla, USA, 1998.

[23] L. J. Malvar and C. A. Ross, "Review of strain rate effects for concrete in tension," *ACI Materials Journal*, vol. 95, no. 6, pp. 735–739, 1998.

[24] Comite Euro-International du beton, *CEB-FIP Model Code 1990*, Wiltshire Redwood Books, Trowbridge, UK, 1993.

[25] I. Rohr, H. Nahme, and K. Thoma, "Material characterization and constitutive modelling of ductile high strength steel for a wide range of strain rates," *International Journal of Impact Engineering*, vol. 31, no. 4, pp. 401–433, 2005.

[26] F. Lin, X. L. Gu, X. X. Kuang, and X. J. Yin, "Constitutive models for reinforcing steel bars under high strain rates," *Journal of Building Materials*, vol. 11, no. 1, pp. 14–20, 2008.

Hindawi Publishing Corporation
Journal of Applied Mathematics
Volume 2014, Article ID 983204, 10 pages
http://dx.doi.org/10.1155/2014/983204

Research Article

Formal Analysis of Fairness for Optimistic Multiparty Contract Signing Protocol

Xiaoru Li, Xiaohong Li, Guangquan Xu, Jing Hu, and Zhiyong Feng

School of Computer Science and Technology, Tianjin University, Tianjin 300072, China

Correspondence should be addressed to Xiaohong Li; xiaohongli@tju.edu.cn

Received 13 February 2014; Accepted 20 April 2014; Published 18 June 2014

Academic Editor: Guiming Luo

Optimistic multiparty contract signing (OMPCS) protocols are proposed for exchanging multiparty digital signatures in a contract. Compared with general two-party exchanging protocols, such protocols are more complicated, because the number of protocol messages and states increases considerably when signatories increase. Moreover, fairness property in such protocols requires protection from each signatory rather than from an external hostile agent. It thus presents a challenge for formal verification. In our analysis, we employ and combine the strength of extended modeling language CSP# and linear temporal logic (LTL) to verify the fairness of OMPCS protocols. Furthermore, for solving or mitigating the state space explosion problem, we set a state reduction algorithm which can decrease the redundant states properly and reduce the time and space complexity greatly. Finally, this paper illustrates the feasibility of our approach by analyzing the GM and CKS protocols, and several fairness flaws have been found in certain computation times.

1. Introduction

The first optimistic multiparty contract signing protocol (OMPCS) was designed by Asokan et al. in 1997 [1]. The goal of this protocol is for all signatories to send signatures on a preagreed-upon contract text to all others and for every signatory to obtain all other signatories' signatures on this contract. With the asymmetric structure, one partner must first send out his signature to the others; thus the fairness of such protocols is hard to guarantee. One method to solve the problem is to use a Trust Third Party (TTP) as an intermediary [2], but this method is not efficient and the TTP becomes a bottleneck as all signatories have to communicate with it. The other method is to design the so-called optimistic multiparty contract signing protocol which has the idea that only when an unfairness problem arises the TTP intervenes [1]. In 2009, Mauw et al. proposed the notion abort-chaining attacks and analyzed the message complexity of OMPCS protocols [3]. Resolve-impossibility which means that it is impossible to define a trusted party protocol for a special OMPCS protocol was presented [4].

Some specific properties such as fairness, timeliness, and abuse-freeness should be satisfied in OMPCS protocols. Asokan defines a fairness system as that if a player behaves correctly, the other players will not gain any advantage over the correctly behaving player, and he divides fairness into strong fairness and weak fairness. Strong fairness means that, when the protocol has completed, A has B's item, or B has gained no additional information about A's item, and vice versa. Here, the assumed "item" means the signed contract, and "additional information" means information which can be obtained from the signed contract. However, weak fairness means that either strong fairness is achieved or a correctly behaving node can prove to an arbiter that an unfair situation has occurred [5].

Although not so much attention has been paid to optimistic multiparty contract signing protocols, there have been several researchers who did certain remarkable work in the weak fairness verification field and got some achievements. The pioneer work can be traced to Chadha et al. [6]. Based on the alternation transition system (ATS) and symbolic tool Mocha, GM protocol was verified to be unfair when the number of signatories is four. Those authors also presented a

CKS protocol and verified that the fairness can be guaranteed in the case of four. Then such method was used in [7] to analyze the fairness of MRT [8] and MR [9] protocols. The MRT protocol failed to satisfy fairness when the number of participators is three, and no flaw has been revealed in the MR protocol.

As an analyzing method, the strand space model has lots of magnificent qualities such as intuitiveness and strictness. In [10], Mukhamedov et al. used a strand space model to describe the GM protocol and found a flaw. Theorem proving method uses formulae to verify the correctness of protocols. The CKS protocol was shown to have flaws in [11] by using the Isabelle/HOL theorem proving machine.

This paper describes a precise and nature formulation of the desired weak fairness and universal OMPCS protocol models that includes multisignatories and contract texts promises simultaneously, formalizes protocol arithmetic, and considers composite attackers of the dishonest signatory. Besides, to mitigate the state explosion problem, a state reduction algorithm is proposed, and conversions between processes are supported, all of such actions can help our analysis to reach cases of more signatories. Furthermore, a visual attacking trace will be given when there exist error traces.

The paper is structured as follows. Section 2 defines the notion of OMPCS protocols and weak fairness. The CSP# language and the LTL logic are briefly introduced in Section 3. This section also explains how to build models for OMPCS protocols and how to express the weak fairness property. In the next section, we apply our novel method to the analysis of two typical OMPCS protocols GM and CKS. Finally, conclusions and future work are summarized.

2. Definitions

Definition 1 (optimistic multiparty contract signing). A protocol for at least n ($n \geq 2$) signatories $P_1, \ldots, P_i, \ldots P_j, \ldots P_n$ to sign a contract m over a Trust Third Party is called a multiparty contract signing (MPCS) protocol. If all signatories are honest, the protocol terminates without T ever sending or receiving any messages; it is called an optimistic multiparty contract signing (OMPCS) protocol. An OMPCS protocol consists of three subprotocols: main subprotocol, recovery subprotocol, and abort subprotocol. The main subprotocol for n signers is divided into n-levels, which can be described recursively. The recovery and abort subprotocols are used to contact T when something goes wrong.

In an OMPCS protocol, every signer has a public and private key pair and can make a digital signature with the private key. $\mathrm{PCS}_{P_1}(m, P_2, T)$ (T is short for TTP) denotes the promise signed by P_1 and sent to P_2, and [10]

(a) $\mathrm{PCS}_{P_1}(m, P_2, T)$ is generated by P_1;

(b) P_2 can forge $\mathrm{PCS}_{P_1}(m, P_2, T)$ but can be distinguished by P_1, P_2, and T;

(c) only P_1 and T can transfer $\mathrm{PCS}_{P_1}(m, P_2, T)$ into a digital signature $S_{P_1}(m)$ which can be universally verified.

Definition 2 (weak fairness). For a protocol Γ, a contract m, and signers P_i and P_j with each one's signed contract $S_{P_i}(m)$ and $S_{P_j}(m)$, we call the protocol Γ which satisfies weak fairness if and only if we get one of the following conditions:

(a) when the protocol is terminated, both P_i and P_j have not received $S_{P_j}(m)$ and $S_{P_i}(m)$ from each other

(b) or when the protocol is terminated, both P_i and P_j have received $S_{P_j}(m)$ and $S_{P_i}(m)$ from each other.

3. Formal Method

Unlike classical security protocols, besides the security property, the OMPCS protocols also need to guarantee fairness between participants. For verifying such property, this paper described an innovative method and Figure 1 shows the structure of it.

3.1. Assumption. Considering the general conditions of OMPCS protocol, assumptions for our fairness analyzing method are as follows.

(a) Channel assumption (lower-case): channels between participants are not trusty; that is to say, the transmitting messages may be delayed and lost. However, channels between participants and the TTP are really reliable; that is, although the transmitting messages may be delayed, they will reach the destination in the limited time.

(b) Participants of protocols: participants may not be honest, but there is at least one honest among a number of participants. The TTP is always honest, and the honest one will follow protocols. A dishonest one can be legal user, possessing his/her own public and private keys.

(c) Attacker: the cryptography is assumed to be perfect, and external hostile agent is assumed not existing for fairness property in OMPCS protocols requires protection from each signatory rather than from external attackers. All of the assumption can let us only concentrate on the structure of the protocols.

3.2. System Variables. CSP# was proposed as an extension of CSP; it combines high-level modeling operators with shared variables and low-level programming constructs. The idea is to treat sequential terminating programs as atomic events [12].

The CSP# model of an OMPCS protocol consists of participants (honest or dishonest), TTP and the communication channel. CSP# mathematical signals were applied to describe the protocols' participants and the components which combine participants together. The syntax of basic operators is listed in Table 1. Where P, Q are processes, e is a name representing an event with an optional sequential program prog, cond is a boolean formula, X is a set of events names, ch is a channel, exp is an expression, and x is a variable.

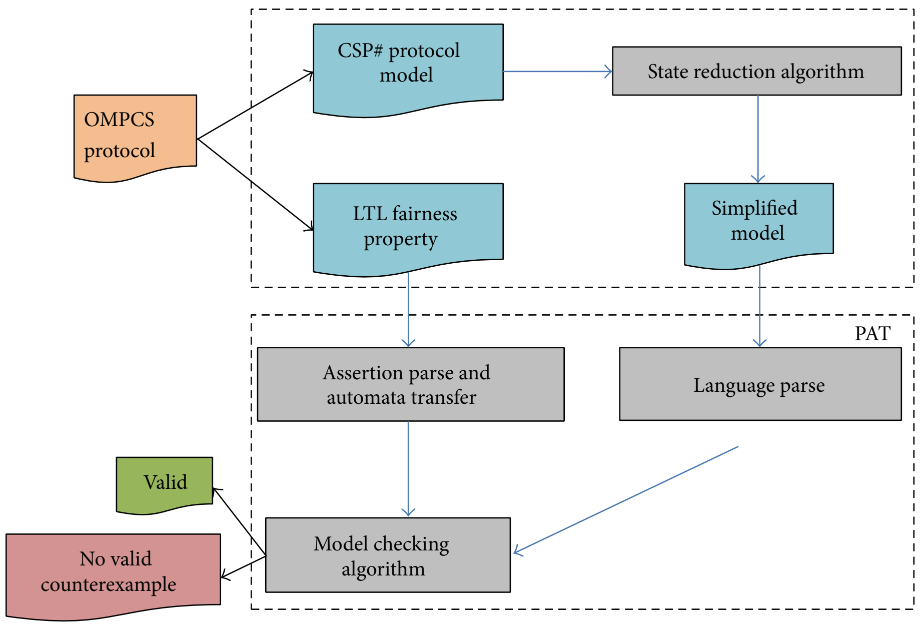


Figure 1: Structure of the method.

Table 1: Syntax of the CSP#.

Stop	Processes do nothing
Skip	Successful termination
$P\,[\ ]\,Q$	General choice
$P\,[*]$	External choice
$P\,\langle\ \rangle\,Q$	Internal choice
$P \mid\mid\mid Q$	Interleaving
$P \mid\mid Q$	Parallel composition
If (cond)$\{P\}$ else $\{Q\}$	Conditional choice
$e\{prog\} \rightarrow P$	Event prefixing
$P \setminus X$	Hiding all occurrences of event in X
$[cond]P$	Guarded process
$P; Q$	Sequential composition
$ch!exp \rightarrow P$	Channel output
$ch?x \rightarrow P$	Channel input

Under the modeling language CSP#, the system variables of protocol model are defined in Table 2. Each sending message between signers is modeled using a variable, initialized to 0 and set as k when the successfully sent message level is k. Each sending message between signers and TTP is modeled using a boolean variable, initialized to false and set true when message was sent.

3.3. Modeling Protocol. The semantic model is based on the labeled transition system (LTS). An LTS is a three-tuple $L = (S, S_0, R)$.

(i) S is set of states. A state can be described by giving value for all valuations in the CSP# model.

(ii) $S_0 \in S$ is the set of initial states.

(iii) $R : S \times \sum\tau \times S$ is the transition relation, $\sum\tau$ is a set of all events, $\sum *$ is a set of all traces, and a trace is a sequence of events. For every state $s \in S$, there is a state $s' \in S$ such that $(s, s') \in R$. R is process expressions in our CSP# protocol model.

A CSP# model configuration is composited of two components (V, P), where V is a function mapping a variable name to its value and P is a process expression. Then we can get the LTS $= (S, \text{init}, \rightarrow)$, where S is the set of reachable system configurations, init is the initial configuration (V, P), and $\rightarrow$ is a transition relation.

For every signer in protocol, we use a process to describe $P_i H_$process, which describes how honest signer P_i behaves and the dishonest P_j has a corresponding process $P_j_$process. $T_$process is defined to model T which represents the processes of recovery and abort subprotocols. The protocol is then described as formula (1); P is the set of signatories:

$$\text{sys} = P_i H\_\text{Process} \mid\mid\mid P_j\_\text{process} \quad (i, j \in P). \tag{1}$$

3.4. Modeling Weak Fairness. In order to reason protocol model, fairness assertion is described by LTL (linear temporal logic). LTL was proposed by Pnueli in 1977 [13] and is used to verify computer program logic language.

LTL is defined by assuming that the atomic formulae are state predicates. Formulae are built up in the usual way according to the following grammar. AP is a collection of atomic propositions of LTL:

(a) for all the atomic propositions $\rho \in$ AP, ρ is a LTL formula;

(b) the constants "true" and "false" are both LTL formulae;

(c) if φ, ϕ are two LTL formulae, then $\neg\varphi$, $\varphi \vee \phi$, $\varphi \wedge \phi$, $G\varphi$, and $F\ \varphi$ are all LTL formulae;

TABLE 2: System variables.

$P_r\_i\_j$	$P_r\_i\_j = k$, if P_i has successfully sent the k level promise to P_j
$P_i\_Recovery\_j$ $P_i\_Recovery\_j\_t$ $P_i\_Recovery\_j\_t\_m$	P_i sends recovery requirement message to TTP, j is the max level of message P_j has sent to P_i, t is the max level of message P_t has sent to P_i, and m is the max level of message P_t has sent to P_i, and so on
P_i_stop	P_i quits the protocol
$P_i\_contacted\_T$	P_i has sent recovery or abort requirement messages to TTP
P_i_Sj	P_i has successfully received the P_j signed contract
$P_i\_AbortTaken$	P_i sends abort requirement message to TTP
$T\_Respondi$	TTP has responded to the requirement of P_i
$T\_Recovery\_send\_P_i$	TTP sends recovery message to P_i
$T\_Abort\_send\_P_i$	TTP sends abort message to P_i
T_Fi	P_i has not sent abort message and TTP will force him/her to abort in certain situation
T_Si	P_i has sent abort message
T_hi	The highest level P_i has sent to higher signer before it contacts TTP
T_li	The lowest level P_i has sent to lower signer before it contacts TTP
T_ki	The highest level P_i has received from P_j $(i < j)$ The highest level P_i has received from all signers m, $(m, j < i)$

(d) every LTL formula can be built up by using finite times of the above formation rules.

Here φ and ϕ are two formulae, G reads as "global" (also can be written as []), and $G\varphi$ means that event φ has to hold on the entire subsequent path. F reads as "finally" (also can be written as <>), $F\varphi$ means that event φ eventually has to hold (somewhere on the subsequent path).

According to the fairness definition and LTL logic, we assume that only signer P_i is honest, and the description of fairness of signer P_i is given in formula (2). It means that, when the protocol is terminated, if there is a reachable trace τ in which a signer in $P_1, \dots, P_{i-1}, P_{i+1}, \dots, P_n$ can receive P_i's digital signature $S_{P_i}(m)$, then there must exist τ' which is reachable from τ and can make P_i receive the digital signature of $P_1, \dots, P_{i-1}, P_{i+1}, \dots, P_n$:

$$\begin{aligned}\langle\rangle\,[]\,\Big(\big(&P_1 \cdot S_{P_i}(m) \vee \cdots \vee P_{i-1} \cdot S_{P_i}(m)\\ &\vee P_{i+1} \cdot S_{P_i}(m) \vee \cdots \vee P_n \cdot S_{P_i}(m)\big)\\ &\longrightarrow []\,\langle\rangle\,\big(P_i \cdot S_{P_1}(m) \wedge \cdots \wedge P_i \cdot S_{P_{i-1}}(m)\\ &\qquad \wedge P_i \cdot S_{P_{i+1}}(m) \wedge \cdots \wedge P_i \cdot S_{P_n}(m)\big)\Big). \end{aligned} \quad (2)$$

3.5. State Reduction Algorithm. The model checking method has many advantages, such as, high level automation and exact description ability. But the idea of this method is based on exhausted state space searching. When we use it to verify some concurrent systems, the state space may be increased exponentially. That is the space explosion problem. Towards the space explosion problem, a detailed state reduction algorithm is demonstrated below and the algorithm is shown in Algorithm 1.

(a) Deleting states that system cannot reach to decrease the searching time.

(b) Deleting states that will absolutely lead to fairness which can make the track of attack trace more conveniently.

(c) Combing transition relations which reach the same next states to compress the state space, that is, $A\{R\}P$ and $B\{R\}P$, we can combine the two transition relations into $(A \vee B)\{R\}P$. Here A, B, and P are states and R is transition function. The proof can be found in "Hoare Logic," rules of consequence.

(d) Letting the dishonest signers send the highest promises to each other, for the transaction between dishonest signers will not influence the fairness of the honest one. Such behaviors can cut down the number of messages between the dishonest signers and then can reduce the state space.

trans is an array which stores the transition relations in the OMPCS model, and every transition relation in trans contains two states as well as a variable; state 1 is the current state, state 2 is the next state, and trans.mark is the weight of states. If a state in a protocol model has been reached for m times, the value of mark for such state is m. The variable exishon(i) is a function to judge whether, after P_i contacting T, there also exists an honest signer who has no contract T. validate(trans[i]) = true means that the transition relation trans[i] contracted T, and T sent a recovery message to reply to it; otherwise, T sent an abort message.

Line 1 to line 2 are corresponding to (a) and (b), according to the recovery subprotocols in OMPCS; if validate = true and the honest signer have not contracted TTP before, this protocol will be fair. Lines 4 to 7 mean (c); if trans[i] and trans[j] have the same next state (state2), trans[i].state1 will be combined with trans[j].state1, and trans[i] will be deleted.

```
Procedure StateReduction(trans)
(1)  for i ← 1 to length(trans)
(2)    do if (vaildate(trans[i]) = 1 && exishon(i)))||(trans[i].mark = 0)
(3)          then delete(trans[i])
(4)        for j ← 1 to length(trans)
(5)            do if (trans[i].state2 = trans[j].state2) && (i! = j)
(6)                then trans[j].state1 ← trans[i]state1 + trans[j].state1
(7)                     delete(trans[i])
(8)  for i ← 1 to n
(9)    do for j ← 1 to n
(10)           do if (pi, pj ∈ dis) && (i! = j)
(11)               then p_i_j ← n, p_j_i ← n
(12) end
```

Algorithm 1: The state reduction algorithm.

```
Modeling the behaviors of honest P1 in the four-party GM main subprotocol
P1H_process()=
......
//(1) honest P1 sends 1-level promises to P2
[][!P1_stop && !P1_contacted_T && Pr_1_4_L==0 && Pr_1_3_L==0 && Pr_1_2_L==0 && Pr_2_1_L==1 && Pr_3_1_L==1 &&
Pr_4_1_L==1]P1sendsp21{Pr_1_2_L=1;}->P1H_process()
//(2) honest P1 sends recovery requirement to T
[] [!P1_stop && !P1_contacted_T && Pr_1_4_L==0 && Pr_1_3_L==0 && Pr_1_2_L==1 && Pr_2_1_L==1 && Pr_4_1_L==1 &&
Pr_3_1_L==1]p1recovery111{P1_contacted_T=true; P1_Recovery_1_1_1=true;}->T_process()
Modeling the behaviors of dishonest P2 in the four-party GM main subprotocol
P2_process()=
......
//(3)dishonest P2 sends 1-level promise to P1
[] [!P2_stop && Pr_2_1_L<1] P2sendP32 {Pr_2_1_L=1;}->P2_process()
//(4)dishonest P2 sends recovery requirement to T
[] [!P2_stop && Pr_4_2_L==1 && Pr_3_2_L==1 && Pr_1_2_L==1]P2recovery111{P2_Recovery_1_1_1=true;}
->T_process()
System definition
sys1H|=P1H_process() ||| P2_process() ||| P3_process() |||P4_process();
```

Algorithm 2: Modeling of four-party GM main subprotocol.

```
T_process()=
Modeling of the four-party GM abort subprotocol
......
//(1)T agrees with the abort requirement from P2
[][!T_Respond2 && P2_Abort_Send && !T_Validated && ( T_S4 || T_S3)]TabortP21{T_S2=true; T_Abort_Send_P2=true;
T_Respond2=true;}->P2H_process()
//(2)T refuses the abort requirement from P2
[][!T_Respond2 && P2_Abort_Send && T_Validated ]TabortP22{T_S2=true; T_Recovery_Send_P2=true; T_Respond2=true;}-
>P2H_process()
Modeling of the four-party GM recovery subprotocol
......
//(3)T agrees with the recovery requirement from P3
[] [P3_Recovery_1_3_3 && !T_Respond4 && !T_Respond3 && !T_Respond2 && !T_Respond1]TreocveryP31
{T_Recovery_Send_P3=true; T_Respond3=true; T_Validated=true;}->P3_process()
//(4)T refuses the recovery requirement from P3
[] [P3_Recovery_1_3_3 && !T_Respond3 && (!T_Respond4 || !T_Respond3 || !T_Respond2 || !T_Respond1)&&
!T_Validated && T_S4]TreocveryP312{T_F1=true;T_F2=true;T_S3=true;T_Abort_Send_P3=true;
T_Respond3=true}->P3_process()
```

Algorithm 3: Modeling of four-party GM abort and recovery subprotocols.

```
sys1H= P1H_process() ||| P2_process() ||| P3_process() |||P4_process();
#define goals1 (P4_S1 || P3_S1 || P2_S1);
#define goals2 (P1_S2 && P1_S3 && P1_S4);
#assert sys1H |= G ((goals1) -> F(goals2));
```

Algorithm 4: Modeling fairness of P_1.

```
T_process()=
Modeling of the five-party CKS abort subprotocol
......
//(1)T agrees with the abort requirement from P2
[][!T_Respond2 && P2_Abort_Send && !T_Validated ]TP2abort1{T_S2=true; T_Abort_Send_P2=true;
T_Respond2=true; T_h2=0; T_l2=1}->P2H_process()
//(2)T refuses the abort requirement from P2
[][!T_Respond2 && P2_Abort_Send && T_Validated ]TP2abort2{T_S2=true; T_Recovery_Send_P2=true;
T_Respond2=true}->P2H_process()
Modeling of the five-party CKS recovery subprotocol
......
//(3)T agrees with the recovery requirement from P3
[][P3_recovery_1_1_3_3 && !T_Respond5 && !T_Respond4 && !T_Respond3 && !T_Respond2 && !
T_Respond1] P3reco1{T_Recovery_Send_P3=true;T_Respond3=true; T_Validated=true;}->P3_process()
//(4)T refuses the recovery requirement from P3
[][P3_recovery_1_1_3_3 && !T_Respond3 && (T_Respond5||T_Respond4||T_Respond3||T_Respond2||
T_Respond1)&& !T_Validted && ((T_S5 && T_l5>0 ||T_S4 && T_l4>0 || T_S2 && T_h2>2||T_S1 &&
T_h1>2 ))]P3reco3{T_Respond3=true;T_S3=true;T_Abort_Send_P3=true;T_h3=3;T_l3=3}->P3_process()
```

Algorithm 5: Modeling of five-party CKS abort and recovery subprotocols.

From lines 8 to 11, n is the number of signers, i, j here are the unique ID for signatories, and dis are the sets for the dishonest signatories. In line 10, if P_i and P_j are dishonest signatories, then they will send the highest promise to each other and it is the pseudocode of (d).

4. Case Study

As an OMPCS protocol, the GM protocol was proposed by Garay and Mackenzie in [14]; then in [5] Asokan modified it and proposed the CKS protocol. In this section, the two protocols are modeled and analyzed on the platform PAT [15] and fairness flaws have been discovered.

4.1. GM and CKS. Each of the GM and CKS protocols has three subprotocols: main subprotocol, recovery subprotocol, and abort subprotocol. Such protocols use zero-knowledge primitives, private contract signatures [14]. The main subprotocol of the two protocols is the same, and major changes are in the recovery and abort subprotocols.

The main subprotocol for n signers is divided into n-level recursions with n-level promises. P_1 sends i-level promise to P_2 which can be denoted as $\mathrm{PCS}_{P_1}((m,i),P_2,T)$. The third party is T; if there is nothing wrong in the execution of main subprotocol, T will not be invoked. Conversely, requirement messages will be sent by signers to T to guarantee fairness. For P_i to abort, it will send the abort message to T; for P_i to recover, it will send the corresponding recovery message. The messages are designed so that T can infer the promises that an honest signer would have sent when it launched the recovery subprotocol. For lack of space, we will not describe the two protocols and the details can be found in [6, 14].

4.2. Modeling GM Protocol. We have modeled and analyzed the GM protocol for three cases, and here we take the case of four parties for example. Figure 2 describes the communications between signers, third party, and channels. We assume that the honest signer is P_1, P_2, P_3, and P_4 are dishonest, and P_2, P_3, and P_4 can collude to cheat P_1. P_1H_process() was defined as the behavior of P_1 and P_2_process(), P_3_process(), P_4_process(), and T_process() as the behaviors of P_2, P_3, P_4, and T. After doing that, the GM protocol was modeled as a parallel system called "sys1H."

4.2.1. Modeling GM Main Subprotocol. Main subprotocol is executed when signers exchange their promises. When $n = 4$, the main subprotocol has 4 levels (see the OMPCS definition in Section 2) recursions. We use integer variables to describe promises between signers, and boolean variables represent messages between signers and T. The details of the main subprotocol model are shown in Algorithm 2.

The honest P_1 mainly performs two kinds of actions in the main subprotocol, which includes sending promises to other signers and sending requirement to T. They are described in step (1) and step (2). Step (1) models the action of sending 1-level promise, in which we use boolean variables,

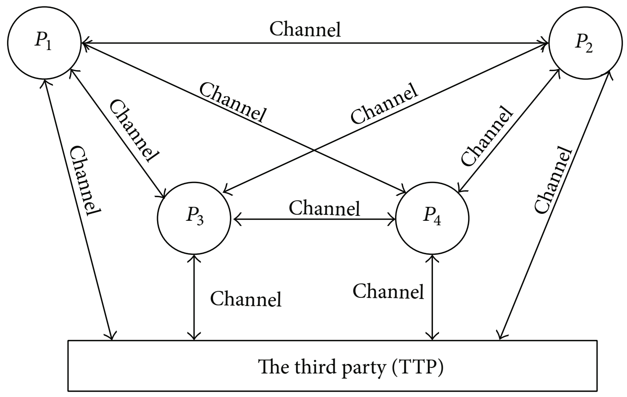


Figure 2: Communications between signers, third party, and channels.

such as Pr_1_3_L, to represent the promises exchanging. Setting Pr_1_2_L = 1 means P_1 has successfully sent 1-level promise to P_2. Step (2) says that if P_1 has not received the correct promises, he can set P_2_Recovery_1_1_1 as true, which represents the action of sending out recovery requirement. Step (3) and Step (4) describe the malicious behaviors of dishonest P_2. Step (3) models that P_2 can send 1-level promise to P_1 and Step (4) specifies that P_2 can send recovery requirement to T at a relatively relaxing condition.

4.2.2. Modeling GM Abort and Recovery Subprotocol. Algorithm 3 models the actions of the T, that is, the abort and recovery subprotocols. T is a special player that has to be modeled in a particular way. The definition and grammar are the same as the main subprotocol. T_process() maintains two sets, $S(m)$ and $F(m)$, which are initialized as empty. When P_i sends abort message to T, $S(m)$ is set to be $S(m) = S(m) \cup \{i\}$. Elements inside $F(m)$ are those signers who have not sent abort message to T. Based on the values of $S(m)$ and $F(m)$, the recovery or abort decision will be made by T.

The actions of T can be divided into two parts, the first part describes how T deals with abort request from P_2. T sends out abort token to P_2 if the status is that T_Validated is false and (T_$S4$ || T_$S3$) is true. However, if T_Validated is true, this means the recovery message has already been sent. Then the abort request is to be refused. Part two models the behaviors of dealing with recovery requests from P_3; if T_Validated is true, the recovery message will be sent. However, conversely, T will make decisions based on the current conditions.

4.3. Modeling GM Fairness. The fairness of P_1 can be divided into two parts. The first is that when GM protocol is finished, if P_1 has not received the contract signed by other signers, then every other signer also will not receive the contract signed by P_1. The remainder is that if any other signers have received the contract signed by P_1, then P_1 must have received the contract signed by other signers as well. So the LTL modeling of P_1's fairness is illustrated in Algorithm 4.

Goal 1 means that at least one signer in P_2, P_3, and P_4 has received the contract signed by P_1. Meanwhile, goal 2 means that P_1 has received the contract signed by P_2, P_3, and P_4. The fairness of P_1 can be described as sys1H| = G((goals 1) $\rightarrow$ F(goals 2)), which means that, for all the traces of GM modeling system, if there is one trace to make one of P_2, P_3, and P_4 receive the signed contract from P_1, there must exist another trace that can guarantee that P_1 receives the signed contract from P_2, P_3, and P_4.

4.4. Modeling CKS Protocol. The model of CKS main subprotocol and the description of fairness remain the same with the GM protocol, so this section will concentrate on the abort and recovery part. The main difference is that T, when presented with a recovery request, overturns its abort decision if and only if T can infer dishonesty on the part of each of the signers that contracted T in the past. Compared with GM, T in CKS also maintains two integer variables, but the difference is that the variables have different meanings than that in GM.

We modeled cases of four and five signers of the CKS protocol and a fairness flaw was found in the case of five. For the sake of contrastive analysis, specifics of P_2 abort request and P_3_Recovery_1_1_3_3 recovery requirements are shown in Algorithm 5.

4.5. State Reduction. When modeling GM and CKS protocol, we mitigate the state space explosion problem based on the algorithm proposed in Section 3.5. The detailed examples are given as below.

(a) Deleting states which cannot be reached: for instance, in the GM protocol, when T is dealing with the requirement P_2_Recovery_1_3_2 from P_2, there is a state which requires that!T_$S3$∧!T_$S4$ and two elements in set {T_$S1$, T_$S3$, T_$S4$} should be true. Then a conclusion can be reached that this state is unreachable, for the state constraint must be (!T_$S3$∧!T_$S4$) ∧ (T_$S3$ ∨ T_$S4$), which is contradictory. Therefore, we can delete this state.

(b) Deleting states which will absolutely lead to fairness: when modeling the GM protocol, if P_4 sends T a recovery requirement P_4_Recovery_3_3_3 in the condition that no one has contacted T before, T will absolutely agree with P_4's requirement, setting T_Validated = 1. It is obvious to ensure that the fairness can be guaranteed for, in the recovery subprotocol, T cannot overturn the recovery decision. Therefore, this state can be ignored when we are searching for attacking trace.

(c) Combining transition relations which have the same next states: in the GM protocol, when T deals with the requirement P_3_Recovery_4_4_4 from P_3there are two transition relations: one of the current states is T_$F3$∧!T_$F2$∧!T_$F4$ and the other is !T_$F3$. However, those two states can reach the same next state; thus, they can be combined into one state ((T_$F3$∧!T_$F2$∧!T_$F4$)∨!T_$F3$).

(d) Letting the dishonest signers send the highest promises to each other: when we model the dishonest signers P_2 and P_3 in the CKS protocol, some of

Table 3: The unfairness trace of the four-party GM protocol.

GM protocol	State 0	State 1	State 2	State 3	State 4
P_4__Recovery_3_3_3	False	False	False	False	True
P_3__abort	False	True	True	True	True
P_2__Recovery_1_1_2	False	False	True	True	True
P_1__Recovery_1_3_3	False	False	False	True	True
T_Abort_Send_P_4	False	False	False	False	False
T_Recovery_Send_P_4	False	False	False	False	True
T_Abort_Send_P_3	False	True	True	True	True
T_Abort_Send_P_2	False	False	True	True	True
T_Abort_Send_P_1	False	False	False	True	True
goals1	False	False	False	False	True
goals2	False	False	False	False	False
Fairness	YES	YES	YES	YES	NO

Table 4: The unfairness trace of the five-party CKS protocol.

Revised GM protocol	State 0	State 1	State 2	State 3	State 4	State 5
P_5_abort	False	True	True	True	True	True
P_4__Recovery_5_4_4_4	False	False	False	True	True	True
P_3__Recovery_6_6_5_5	False	False	False	False	False	True
P_2__Recovery_5_5_5_5	False	False	False	False	True	True
P_1__Recovery_1_4_4_4	False	False	True	True	True	True
T_Abort_Send_P_5	False	True	True	True	True	True
T_Abort_Send_P_4	False	False	False	True	True	True
T_Abort_Send_P_3	False	False	False	False	False	True
T_Abort_Send_P_2	False	False	False	False	True	True
T_Abort_Send_P_1	False	False	True	True	True	True
goals1	False	False	False	False	False	True
goals2	False	False	False	False	False	False
Fairness	YES	YES	YES	YES	YES	NO

the intermediate states such as Pr_2_3_L = 3 and Pr_3_2_L = 2 can be deleted, letting P_2 and P_3 send the highest promise to each other directly, Pr_2_3_L = 4 and Pr_3_2_L = 4.

4.6. Analysis. One feature of the OMPCS protocol is that the number of participants can be varied and the structure is not symmetric. That is why the specification of P_i is different from that of P_j $(i \neq j)$. A number of CSP# models have been written in PAT for different signers for the GM and CKS protocols. We found that GM protocol cannot satisfy fairness for the case of $n \geq 4$, because we can verify that the protocol is not fair in case of four, and, in more signatories cases, if there is only one honest, then the dishonest can collaborate to cheat just as they do in the case of 4. The CKS protocol is not fair for the number of signers $n \geq 5$, the reason is the same for GM.

Table 3 shows one possible error trace for the GM protocol (P_1 is the honest one and $n = 4$). State 0 is the initialized state. In state 1, P_3 contacts T by sending an abort requirement. According to [14], T agrees with this requirement and sends abort message, setting T_$S3$ = true. But the dishonest P_3 continues to execute the main subprotocol. Then P_2 contract T with a recovery requirement. T refuses P_2's requirement based on the abort subprotocol with T_$F1$ = true, T_$S2$ = true. However, the dishonest P_2 continues to execute the main subprotocol. In the next state P_1 launches a resolve request to T, however T_$F1$ is true and it indicates that T will refuse P_1's requirement and update T_$S1$ = true. Then the honest signer P_1 quits the main subprotocol. At last, P_4 sends a recovery requirement. For T_$S4$ is false, T overturns the previous decision and agrees with P_4's requirement. Hence the fairness of P_1 are violated. We also found unfairness trace when P_3 or P_2 is the dishonest signer.

Table 4 shows one possible unfairness trace for the CKS protocol (P_3 is the honest one and $n = 5$). State 0 is also the initialized state. In step 1, P_5 sends an abort requirement to T. T agrees with this requirement and sends back an abort message with T_$l5$ = 1, T_$S5$ = true. But the dishonest P_5 continues to execute the protocol. In state 2, P_1 contract T with a recovery requirement. T refuses P_1's requirement and sets T_$h1$ = 4, T_$S1$ = true, the dishonest P_1 continues to execute the main subprotocol. Then P_4 launches a recovery request to T. T refuses P_4's requirement with an

Table 5: Experiment comparison.

	This paper method		Reference [5] method	
	Time used	Memory used	Time used	Memory used
GM protocol				
$n = 2$	0.01	41179	0.050	51228
$n = 3$	10.64	72056	38.72	92595
$n = 4$	505.21	2479154	—	—
CKS protocol				
$n = 2$	0.01	39187	0.047	4568
$n = 3$	8.29	60145	19.54	86347
$n = 4$	552.31	2787336	—	—
$n = 5$	1674.05	3105836	—	—

abort message, setting $T\_l4 = T\_h4 = 5$, $T\_S4 =$ true. In the following states, P_2 send recovery messages to T and T will refuse it and update $T\_l2 = T\_h2 = 5$, $T\_S2 =$ true. P_3 sends its signed contract to P_4 and P_5, but the dishonest P_4 and P_5 quits the protocol and P_3 contracts T with recovery requirement.

However, according to [5] and the recovery subprotocol, T computes the value of T_hi and T_ki and finds $(T\_S2) \wedge (T\_k2 \leq T\_h2)$, and then it makes a decision that it will not overturn the previous decision. Finally, T sends an abort message to P_3; thus, the fairness of P_3 is not guaranteed. The similar error traces can also be found when P_4, P_2, or P_1 is the dishonest one.

4.7. Experiment Comparison. In this paper, the fairness for four-party GM protocol and five-party CKS protocol has been verified. There are certain advantages of our method, such as less time and space complexity, more precise semantic, higher degree of automation, and more detailed results. The time and space consuming comparison between our method and the method in [5] is showed in Table 5; the unit for time is second and for memory is KB.

We use a common environment for all the tests discussed in this section; the hardware environment for the two methods is the same. For [5], the model checking platform is cmocha and the operating system is Ubuntu. For our method, the model checking platform is PAT3, and the operating system is Windows 7.

As Table 5 demonstrated, although the problem of protocol fairness verification is a NP problem and the states and time are expected to increase exponentially in theory, our method can get the verification result in certain times in the three and four participants cases of the CKS protocol. Therefore, we can get a conclusion that the reduction algorithm did a good performance as no error trace was found in such cases, and every state in the state space was visited. On the contrary, the method in [5] consumed more time and memory. Specifically, in the four-party occasion, in our experiment environment, this method cannot get verification result in certain hours (we had waited for more than 12 hours). Moreover, this method cannot give explicitly a number of visited states and fairness counterexamples. The main reason is that our method has less system states, higher states compression ratio, and the trace back-track algorithm.

5. Conclusion and Future Work

Based on modeling language CSP# and linear temporal logic, an efficient method which aims to analyze the fairness of optimistic multiparty contract signing protocols is presented in this paper. In order to demonstrate the feasibility of our method, two examples of the GM and CKS protocol have been described and several fairness attacking traces have been found. Comparisons also have been made in this paper between our method and other traditional methods. And the result shows that our method has certain strengths such as higher automation, less time and space complexity, visualized attacking trace, and better utility.

There is no explicit model for any cryptographic primitives in this paper and the attack model is weaker than the original paper. The main challenge is to verify the OMPCS protocols fairness in a more general condition which accounts for cryptographic and a more relaxed communication model. In our next work, we plan to extend the automatic cryptographic protocol verifier ProVerif [16] to suit our fairness verification method.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgment

This work was supported in part by the National Science Foundation of China (nos. 91118003, 61272106, and 61003080).

References

[1] N. Asokan, M. Schunter, and M. Waidner, "Optimistic protocols for fair exchange," in *Proceedings of the 4th ACM Conference on Computer and Communications Security*, pp. 6–17, New York, NY, USA, April 1997.

[2] S. Even and Y. Yacobi, "Relations among public key signatures systems," Tech. Rep. 175, Technion, Haifa, Israel, 1980.

[3] S. Mauw, S. Radomirović, and M. T. Dashti, "Minimal message complexity of asynchronous multi-party contract signing," in *Proceedings of the 22nd IEEE Computer Security Foundations Symposium (CSF '09)*, pp. 13–25, IEEE Computer Society Press, July 2009.

[4] A. Mukhamedov and M. D. Ryan, "Resolve-impossibility for a contract-signing protocol," in *Proceedings of the 19th IEEE Computer Security Foundations Workshop (CSFW '06)*, pp. 167–173, July 2006.

[5] N. Asokan, *Fairness in electronic commerce [Ph.D. thesis]*, University of Waterloo, 1998.

[6] R. Chadha, S. Kremer, and A. Scedrov, "Formal analysis of multi-party contract signing," in *Proceedings of the 17th IEEE Computer Security Foundations Workshop (CSFW '04)*, pp. 266–279, IEEE Computer Society Press, Washington, DC, USA, June 2004.

[7] Y. Zhang, C. Zhang, J. Pang, and S. Mauw, "Game-based verification of contract signing protocols with minimal messages," *Innovations in Systems and Software Engineering*, vol. 8, no. 2, pp. 111–124, 2012.

[8] A. Mukhamedov and M. D. Ryan, "Fair multi-party contract signing using private contract signatures," *Information and Computation*, vol. 206, no. 2–4, pp. 272–290, 2008.

[9] S. Mauw, S. Radomirović, and M. T. Dashti, "Minimal message complexity of asynchronous multi-party contract signing," in *Proceedings of the 22nd IEEE Computer Security Foundations Symposium (CSF '09)*, pp. 13–25, IEEE CS, July 2009.

[10] A. Mukhamedov, S. Kremer, and E. Ritter, "Analysis of a multi-party fair exchange protocol and formal proof of correctness in the strand space model," in *Financial Cryptography and Data Security*, vol. 3570 of *Lecture Notes in Computer Science*, pp. 255–269, 2005.

[11] N. Zhang, X. Zhang, and Y. Wang, "Formal analysis of GM multi-party contract signing protocol," in *Proceedings of the 2nd International Conference on Convergent Information Technology (ICCIT '07)*, pp. 1316–1321, IEEE Computer Society Press, November 2007.

[12] J. Sun, Y. Liu, S. D. Jin, and C. Chen, "Integrating specification and programs for system modeling and verification," in *Proceedings of the 3rd IEEE International Symposium on Theoretical Aspects of Software Engineering (TASE '09)*, pp. 127–135, July 2009.

[13] A. Pnueli, "The temporal logic of programs," in *Proceedings of the 18th Annual Symposium on Foundations of Computer Science (FOCS '77)*, pp. 46–57, 1977.

[14] J. A. Garay and P. Mackenzie, "Abuse-free multi-party contract signing," in *Distributed Computing*, vol. 1693 of *Lecture Notes in Computer Science*, pp. 151–165, 1999.

[15] Y. Liu, J. Sun, and J. S. Dong, "PAT 3: an extensible architecture for building multi-domain model checkers," in *Proceedings of the 22nd IEEE International Symposium on Software Reliability Engineering (ISSRE '11)*, pp. 190–199, Hiroshima, Japan, December 2011.

[16] V. Cheval and B. Blanchet, "Proving more observational equivalences with ProVerif," in *Principles of Security and Trust*, vol. 7796 of *Lecture Notes in Computer Science*, pp. 226–246, Springer, 2013.

Hindawi Publishing Corporation
Journal of Applied Mathematics
Volume 2014, Article ID 352152, 20 pages
http://dx.doi.org/10.1155/2014/352152

Research Article

Unified Mathematical Framework for Slicing and Symmetry Reduction over Event Structures

Xinyan Gao,[1] Yingcai Ding,[1] Wenbo Liu,[1] Kaidi Zheng,[1] Siyu Huang,[1] Ning Zhou,[2,3] and Dakui Li[1]

[1] *G&S Labs, School of Software, Dalian University of Technology, Dalian 116620, China*
[2] *School of Electronic and Information Engineering, Lanzhou Jiaotong University, Lanzhou 730070, China*
[3] *School of Computer and Information Technology, Beijing Jiaotong University, Beijing 100044, China*

Correspondence should be addressed to Dakui Li; ldk@dlut.edu.cn

Received 28 January 2014; Accepted 27 April 2014; Published 12 June 2014

Academic Editor: Xiaoyu Song

Nonclassical slicing and symmetry reduction can act as efficient structural abstract methods for pruning state space when dealing with verification problems. In this paper, we mainly address theoretical and algorithmic aspects for nonclassical slicing and symmetry reduction over prime event structures. We propose sliced and symmetric quotient reduction models of event structures and present their corresponding algorithms. To construct the underlying foundation of the proposed methodologies, we introduce strong and weak conflict concepts and a pair of mutually inverse operators and extend permutation group based symmetry notion of event structures. We have established a unified mathematical framework for slicing and symmetry reduction, and further investigated the translation, isomorphism, and equivalence relationship and other related basic facts from a theoretical point of view. The framework may provide useful guidance and theoretical exploration for overcoming verification challenges. This paper also demonstrates their practical applications by two cases.

1. Introduction

Generally, to detect whether a finite execution trace of a distributed program satisfies a given predicate, namely, predicate detection (a kind of verification problems), is a fundamental problem in asynchronous distributed systems. It has applications in many domains such as testing, debugging, and monitoring of distributed programs and it is also a powerful runtime verification method.

Unfortunately, predicate detection is NP complete [1] and suffers from the excessive size of the state space and the state explosion problem—the number of possible global states of the program increases exponentially owing to simple combination.

To deal with this problem, several useful reduction techniques have been suggested in succession for reducing the state space in recent years, such as partial order reduction and symmetric reduction methods [2–4].

On the one hand, the basic observation is that many distributed or concurrent systems exhibit a certain degree of symmetry, for example, a system composed of identical or isomorphic components whose identities are interchangeable from a verification point of view. This kind of structural symmetry in the system is also reflected in the full state space of the system. The main idea behind the symmetry reduction method is to figure out this symmetry and obtain a condensed state space which is typically much smaller than the full state space, but from which the same kind of properties of the system can be derived without unfolding the condensed state space to the full state space. Thus, it can be used to verify any property of the original model.

On the other hand, a slice of a system with respect to a criterion is a subsystem that only contains all the states of the original system that satisfy this specification. The advantage of this technique lies in the fact that the detection is performed only on the small part of the global state space which is of interest. In many cases, the slice is exponentially smaller than the original. In order to tangle predicate detection problem, nonclassical slicing technique, named computation slicing, as an abstraction mechanism, inspired by the classical program slicing of Weiser [5, 6], was first proposed by Garg and Mittal [7].

For the majority predicate classes, the computation slicing algorithm has polynomial-time complexity and gains exponential reduction of state spaces. Computation slicing has been proved to be an efficient technique for pruning state space of predicate detection in distributed computation. Moreover, it has also been successfully applied to solve the problems of temporal properties verification in transaction level hardware descriptions such as PCI local bus protocol and the MSI (modified shared invalid) cache coherence protocol [6] in SoC (system on chip) systems and so forth.

Due to the restriction of partial order execution trace model [5, 8], this approach has some limitations. Firstly, it is a runtime checking method and only checks a single partial execution trace once. It is not easy to obtain 100% path coverage even though this detection is performed multiple times. Thus, it is not suitable for exhaustive analysis by reasoning about all possible execution of the system model. Secondly, its underlying model is partially ordered set and it is not expressive enough to handle these models with explicit choice structures or conflicts. Because all the runtime traces do not contain any conflict information, it is not convenient to analyze the system under construction statically.

In this paper, we extend the notion of computation slicing from partial order traces to prime event structures with conflict. We propose a more general event structure slicing notion and a complete mathematical theoretical framework for computing the event structure slices.

The main idea is that a prime event structure can be viewed as such a system model consisting of several conflict-free substructures. These substructures themselves are in mutual weak conflict. Any of such conflict-free event substructures of a prime event structure acts as a partial order execution trace which can be sliced by traditional computation slicing algorithm. Based on this idea, we propose a partition approach to decompose a prime event structure into a group of conflict-free substructures equivalently. Each of these substructures can be sliced with respect to a given slicing criterion by the existing slicing algorithm and we can get a set of the sliced substructures. We have proved that these sliced results can be composed together and yield a new prime event structure by a so-called weak choice composition operation. We have shown that the newly generated prime event structure is the slicing result of the original prime event structure. Meanwhile, based on above partition, we can detect structural symmetry property and make symmetric reduction on each substructure of the original system. In additional, we also investigate the relationship between the symmetric reduction model and the original one.

The main contribution of our work can be summarized as follows. We introduced the slicing notion into the area ofevent structure and extended nonclassical computation slicing with conflict. We also proposed a unified mathematical framework as a common theory basis for event structure slicing and symmetry reduction. We also made a comparison between our event structure slicing and the traditional computation slicing and demonstrated the mathematical aspects of this framework.

The rest of this paper is structured as follows. Related work is discussed in Section 2. Section 3 introduces the notion of event structure and other basic definitions. Section 4 describes two core operators over event strictures. Slicing reduction derived from computation slicing will be discussed in Section 5. Symmetry reduction theory based on permutation group is reported in Section 6. The overall mathematical framework for event structure slicing and symmetry reduction will be provided in Section 7. In the last section, we make a short summary of our work.

2. Related Work

Regarding the slicing technique, the work in [5, 6] proposed classical program slice idea firstly by Weiser. Given a program and a set of variables, a program slice consists of all statements in the program that may affect the value of the variables in the set at some given point.

During years after the program slice notion was proposed, a lot of work based on this notion had been performed. For example, in 1992, the notion of a slice has been also extended to distributed programs [9]. In 2000, the notion of a nonclassical computation slice, which is very similar to the concept of a program slice, has been proposed. In work [7, 10], computation slice over partial order traces was firstly investigated by Garg and Mittal, de Bakker et al. This computation slice notion is based on partial order traces model, which is a special case of event structure without conflict.

Event structure, as an true concurrency model [11–16], can be taken as an extension of partial order model. In concurrency theory, event structures constitute a major branch of concurrent models. These were initially developed as a link between Petri nets and Scott domain theory [17] and have since been extensively applied as a semantic model for process algebras, for example [18].

All the previous work [7, 8, 19, 20] does not consider the case with conflict. Compared with them, our work is aimed to extend this slicing notion to the area of event structure.

On the other hand, as for symmetry reduction, the use of symmetry to reduce state space has been investigated widely by researchers. Technically speaking, symmetry in event structures [3, 4] is similar to symmetry in model checking [2, 21, 22]. In work [23], a category of event structures with symmetry was introduced and its categorical properties were investigated, while our work is relevant to the structural reduction via symmetry property over event structure model.

In our previous work [24], we have extended this technique to event structure area. In this paper, we will further investigate the common basis for both slicing and symmetry reduction over event structures and provide a unified framework.

3. Event Structure and Basic Definitions

In this section, we will introduce the notion of prime event structure [11, 17, 25, 26] and the basic definitions we use throughout the paper. The prime event structure is firstly defined and other related key notions are introduced. Moreover, we focus on finite prime event structures only.

Definition 1 (prime event structure). A *prime event structure* (over an alphabet $\mathscr{A}$, a set of actions) is a 4-tuple structure $(E, \preceq, \sharp, l)$ with

(i) E, a finite set of events;

(ii) $\preceq \subseteq E \times E$, a partial order, the causality relation, satisfying the principle of finite causes: for all $e \in E$: $\{e' \in E \mid e' \preceq e\}$ is finite and the inverse of $\preceq$ is denoted by $\preceq^{-1}$;

(iii) $\sharp \subseteq E \times E$, the (irreflexive and symmetric) conflict relation, satisfying the principle of conflict inheritance: $\forall d, e, f \in E : d \preceq e \wedge d \sharp f \Rightarrow e \sharp f$;

(iv) $l : E \rightarrow \mathscr{A}$, the action-labelling function.

A prime event structure (for short, an event structure) represents a system in the following way: the action names are activities which the system may perform, an event labelled $a \in \mathscr{A}$ stands for a particular occurrence of an action, $e_a \preceq e_b$ indicates that a cannot occur before b has, and $e_c \sharp e_d$ indicates that actions c and d can never occur together in one run.

The conflict inheritance property states that if an event e is in conflict with some event f, then it is in conflict with all causal successors of f.

From the causality relation, it is not difficult to derive a notion of *causal independence:*

$$e co d \Longleftrightarrow \neg \left(e = d \vee e \preceq d \vee e \preceq^{-1} d \vee e \sharp d\right). \qquad (1)$$

Let $\mathbb{E}$ denote the domain of prime event structures labelled over $\mathscr{A}$ and $\emptyset = (\emptyset, \emptyset, \emptyset, \emptyset)$ stand for the empty event structure. Generally, the components of an event structure $\mathscr{E}$ will be denoted by $E_{\mathscr{E}}$, $\preceq_{\mathscr{E}}$, $\sharp_{\mathscr{E}}$, and $l_{\mathscr{E}}$, respectively. More specifically, $\mathscr{E} = (E_{\mathscr{E}}, \preceq_{\mathscr{E}}, \sharp_{\mathscr{E}}, l_{\mathscr{E}})$. If clear from the context, the index will be omitted; that is, $\mathscr{E} = (E, \preceq, \sharp, l)$ is also a valid form.

Additionally, for $X \subseteq E_{\mathscr{E}}$, the restriction of $\mathscr{E}$ to X can be defined as $\mathscr{E}|_X = (X, \preceq_{\mathscr{E}} \cap (X \times X), \sharp_{\mathscr{E}} \cap (X \times X), l_{\mathscr{E}}|_X)$. Let Succ($e$) denote all causal successors of an event e; that is, $\mathrm{Succ}(e) = \{a \in E_{\mathscr{E}} \mid e \preceq_{\mathscr{E}} a, e \in E_{\mathscr{E}}\}$.

Definition 2 (event substructure). Let $\mathscr{E} = (E_{\mathscr{E}}, \preceq_{\mathscr{E}}, \sharp_{\mathscr{E}}, l_{\mathscr{E}}) \in \mathbb{E}$ and $\mathscr{E}' = (E_{\mathscr{E}'}, \preceq_{\mathscr{E}'}, \sharp_{\mathscr{E}'}, l_{\mathscr{E}'}) \in \mathbb{E}$ be event structures; $\mathscr{E}'$ is called a *substructure* of $\mathscr{E}$ (denoted by $\mathscr{E}' \lhd \mathscr{E}$) if and only if

(i) $E_{\mathscr{E}'} \subseteq E_{\mathscr{E}}$;

(ii) for all $e, e' \in E_{\mathscr{E}}$, $e \sharp_{\mathscr{E}'} e' \Leftrightarrow e, e' \in E_{\mathscr{E}'} \wedge e \sharp_{\mathscr{E}} e'$;

(iii) for all $e, e' \in E_{\mathscr{E}}$, $e \preceq_{\mathscr{E}'} e' \Leftrightarrow e, e' \in E_{\mathscr{E}'} \wedge e \preceq_{\mathscr{E}} e'$.

Definition 3 (conflict-free event structure). An event structure $\mathscr{E} = (E_{\mathscr{E}}, \preceq_{\mathscr{E}}, \sharp_{\mathscr{E}}, l_{\mathscr{E}}) \in \mathbb{E}$ is called *conflict-free event structure* (denoted by $cfES$, for short) if and only if its conflict relation is empty; that is, $\sharp_{\mathscr{E}} = \emptyset$.

Let $\mathbb{F}$ denote the domain of conflict-free prime event structures.

In order to characterize the conflict relationship between two conflict-free event structures (or substructures of a prime event structure), we introduce the following basic definitions: strong conflict, weak conflict, and weak conflict event structure set (for short, weak conflict set).

Definition 4 (strong conflict). Let $\mathscr{F}_1 = (E_{\mathscr{F}_1}, \preceq_{\mathscr{F}_1}, \emptyset, l_{\mathscr{F}_1}) \in \mathbb{F}$ and $\mathscr{F}_2 = (E_{\mathscr{F}_2}, \preceq_{\mathscr{F}_2}, \emptyset, l_{\mathscr{F}_2}) \in \mathbb{F}$. The conflict relation between E_1 ($E_1 \subseteq E_{\mathscr{F}_1}$ and $E_1 \neq \emptyset$) and E_2 ($E_2 \subseteq E_{\mathscr{F}_2}$ and $E_2 \neq \emptyset$) is called *strong conflict* if and only if for all $e \in E_1, f \in E_2 : e \sharp f$, denoted by $E_1 \sharp^s E_2$. $\mathscr{F}_1$ and $\mathscr{F}_2$ are called *strong conflict* if and only if their event sets are in mutually strong conflict, that is, for all $\mathscr{F}_1, \mathscr{F}_2 \in \mathbb{F}$: $\mathscr{F}_1 \sharp^s \mathscr{F}_2 \Leftrightarrow E_{\mathscr{F}_1} \sharp^s E_{\mathscr{F}_2}$, denoted by $\mathscr{F}_1 \sharp^s \mathscr{F}_2$.

More generally, for any $\mathscr{E}_1 \in \mathbb{E}$ and $\mathscr{E}_2 \in \mathbb{E}$, the relation between nonempty E_1' ($\emptyset \neq E_1' \subseteq E_{\mathscr{E}_1}$) and E_2' ($\emptyset \neq E_2' \subseteq E_{\mathscr{E}_2}$) is called *extended strong conflict* if and only if for all $e \in E_1', f \in E_2' : e \sharp f$, denoted by $E_1' \sharp^{xs} E_2'$. That is, each of E_1' is in conflict with each of E_2' and the existence of conflict relation in E_1' or E_2' is allowed.

Definition 5 (weak conflict). Let $\mathscr{F}_1 = (E_{\mathscr{F}_1}, \preceq_{\mathscr{F}_1}, \emptyset, l_{\mathscr{F}_1}) \in \mathbb{F}$ and $\mathscr{F}_2 = (E_{\mathscr{F}_2}, \preceq_{\mathscr{F}_2}, \emptyset, l_{\mathscr{F}_2}) \in \mathbb{F}$. The conflict relation between event sets E_1 ($E_1 \subseteq E_{\mathscr{F}_1}$ and $E_1 \neq \emptyset$) and E_2 ($E_2 \subseteq E_{\mathscr{F}_2}$ and $E_2 \neq \emptyset$) is called *weak conflict* if and only if $\exists e \in E_1, \exists f \in E_2 : e \sharp f$, denoted by $E_1 \sharp^w E_2$. The conflict-free event structures, $\mathscr{F}_1$ and $\mathscr{F}_2$, are called *weak conflict* if and only if their event sets are in weak conflict; that is, for all $\mathscr{F}_1, \mathscr{F}_2 \in \mathbb{F} : \mathscr{F}_1 \sharp^w \mathscr{F}_2 \Leftrightarrow E_{\mathscr{F}_1} \sharp^w E_{\mathscr{F}_2}$, denoted by $\mathscr{F}_1 \sharp^w \mathscr{F}_2$.

Stated in words, it is not that each event of $E_{\mathscr{F}_1}$ is in conflict with each event of $E_{\mathscr{F}_2}$, but there exists at least one conflicting event pair between $E_{\mathscr{F}_1}$ and $E_{\mathscr{F}_2}$.

Basically, according to the previous definitions, strong conflict relation is a special case of weak conflict relation.

Definition 6 (weak conflict set). Let $\mathscr{WF}_n^{cfw} = \{\mathscr{F}_i \in \mathbb{F} \mid i, n \in \mathbf{N}, \mathbf{1} \leq \mathbf{i} \leq \mathbf{n}\}$ over event set $E_{\mathscr{WF}}$, $\mathscr{WF}_n^{cfw}$ is called a *weak conflict set* if and only if $E_{\mathscr{WF}} = \bigcup_{i=1}^{n} E_{\mathscr{F}_i}$ and for all $\mathscr{F}_i, \mathscr{F}_j \in \mathscr{WF}_n^{cfw} : i \neq j \Rightarrow \mathscr{F}_i \sharp^w \mathscr{F}_j$ $(i, j \in N, 1 \leq i \leq n)$.

For convenience, let $wfsetF(\mathscr{WF}_n^{cfw}) = \{E_{\mathscr{F}_i} \mid i \in \mathbf{N}, 1 \leq i \leq n\}$ denote the family of weak conflict event sets.

Definition 7 (maximal conflict-free event substructure). Let $\mathscr{E} = (E_{\mathscr{E}}, \preceq_{\mathscr{E}}, \sharp_{\mathscr{E}}, l_{\mathscr{E}}) \in \mathbb{E}$ be an event structure; any event subset $E' \subseteq E_{\mathscr{E}}$ is called a *maximal conflict-free event subset* (for short, mcfset) of $E_{\mathscr{E}}$ if and only if it satisfies the following:

(1) for all $e, f \in E' : \neg(e \sharp f)$;

(2) for all $c \in (E_{\mathscr{E}} - E') : \exists d \in E' \Rightarrow c \sharp d$.

Its corresponding substructure $\mathscr{E}'$ is called *maximal conflict-free event substructure* of $\mathscr{E}$; that is, $\mathscr{E}' = (E', \preceq_{\mathscr{E}} \cap (E' \times E'), \emptyset, l_{E'})$.

4. Operators over Event Structure

In this section, a pair of mutually inverse operators, cfp (conflict-free partition) and wcc (weak conflict composition), will be introduced and discussed. For any prime event structure $\mathscr{E}$, partition and composition operation over it can be associated via its family of configurations.

4.1. Maximal Conflict-Free Partition. In fact, a prime event structure can be viewed as a system consisting of several substructures, which are conflict-free themselves. Such a conflict-free event substructure of a prime event structure represents a specific possible partial order execution trace via branching or nondeterministic choices. For any prime event structure, it is a certainty that we can get its maximal conflict-free substructures by some kind of conflict-free partition operation according to the characteristics of its conflict relation.

First of all, we give the definition of maximal conflict pattern for an event structure. The notion of maximal conflict pattern can make great contributions to accelerate the process of partition by avoiding unnecessary partition steps. We then provide the key partition algorithm for a prime event structure.

Definition 8 (maximal conflict pattern). Let $\mathscr{E} = (E_{\mathscr{E}}, \leq_{\mathscr{E}}, \sharp_{\mathscr{E}}, l_{\mathscr{E}}) \in \mathbb{E}$; for any $A \subseteq E_{\mathscr{E}}$ and $B \subseteq E_{\mathscr{E}}$, $A\sharp^{xs}B$ is called a *maximal conflict pattern*if and only if for all $A' \subseteq E_{\mathscr{E}}$, for all $B' \subseteq E_{\mathscr{E}} : (A \subset A' \wedge B \subset B') \Rightarrow \neg(A'\sharp^{xs}B')$.

For any prime event structure, we can get these maximal conflict patterns by the following two steps:

(1) casual successors expanding;

(2) conflict pairs merging.

Firstly, due to the conflict inheritance property, we know that if event e is in conflict with event f then their casual successors are also in mutual conflict; that is, $\forall c \in \mathrm{Succ}(e), \forall d \in \mathrm{Succ}(f) : e\sharp f \Rightarrow c\sharp d$.

Let $\{e\}_{\sharp} = \{e\} \cup \mathrm{Succ}(e)$ and $\{f\}_{\sharp} = \{f\} \cup \mathrm{Succ}(f)$; we have that $\{e\}_{\sharp}$ and $\{f\}_{\sharp}$ are in strong conflict if $e\sharp_{\mathscr{E}}f$; namely, $\forall e, f \in E_{\mathscr{E}} : e\sharp_{\mathscr{E}}f \Rightarrow \{e\}_{\sharp}\sharp^{s}\{f\}_{\sharp}$.

For example, for a prime event structure $\mathscr{E}$, if $c\sharp_{\mathscr{E}}d$ and casual relations are $c\preceq_{\mathscr{E}}e_1$, $c\preceq_{\mathscr{E}}e_2$, $d\preceq_{\mathscr{E}}f_1$, $f_1\preceq_{\mathscr{E}}f_2$, and $d\preceq_{\mathscr{E}}f_3$, we then have $\{c, e_1, e_2,\}\sharp^{s}\{d, f_1, f_2, f_3\}$.

We also have that any nonempty subset of $\{e\}_{\sharp}$ and any nonempty subset of $\{e\}_{\sharp}$ are also in strong conflict.

Consider a prime event structure $\mathscr{E} \in \mathbb{E}$ whose conflict relation has l ($l \in \mathbf{N}$) conflict pairs. Expand each conflict relation with its successors according to the conflict inheritance property and we can get full conflict relation pairs: $\{e_1\}_{\sharp}\sharp^{s}\{f_1\}_{\sharp}, \ldots, \{e_l\}_{\sharp}\sharp^{s}\{f_l\}_{\sharp}$; here, $e_i\sharp_{\mathscr{E}}f_i$, $e_i \in E_{\mathscr{E}}$, $f_i \in E_{\mathscr{E}}(i \in \mathbf{N}, 1 \leq i \leq l)$.

Secondly, for such a group of full conflict relation pairs obtained by the above steps, there may exist common elements among some pairs that can be merged together and form a maximal conflict pattern. For example, events e_1, e_2, e_3, and e_4 are mutually in conflict; we have six immediate conflict relation pairs: $\{e_1\}\sharp^{s}\{e_2\}$, $\{e_1\}\sharp^{s}\{e_3\}$, $\{e_1\}\sharp^{s}\{e_4\}$, $\{e_2\}\sharp^{s}\{e_3\}$, $\{e_2\}\sharp^{s}\{e_4\}$, and $\{e_3\}\sharp^{s}\{e_4\}$.

From the principle of permutation, we then have three maximal conflict patterns by merging conflict pairs: $\{e_1\}\sharp^{xs}\{e_2, e_3, e_4\}$, $\{e_2\}\sharp^{xs}\{e_3, e_4\}$, and $\{e_3\}\sharp^{xs}\{e_4\}$. Equivalently, $\{e_4\}\sharp^{xs}\{e_1, e_2, e_3\}$, $\{e_3\}\sharp^{xs}\{e_1, e_2\}$, and $\{e_2\}\sharp^{xs}\{e_1\}$ are also the valid maximal conflict patterns.

Assume that there are $m \in \mathbf{N}$ maximal conflict patterns after expanding and merging which are $\mathscr{I}^{1}_{\sharp_{\mathscr{E}}} : A_1\sharp^{xs}B_1, \ldots, \mathscr{I}^{m}_{\sharp_{\mathscr{E}}} : A_m\sharp^{xs}B_m$, respectively.

Here, $\mathscr{I}^{i}_{\sharp_{\mathscr{E}}} : A_i\sharp^{xs}B_i$ denotes the ith pattern, and $\forall e \in A_i \subseteq E_{\mathscr{E}}, \forall f \in B_i \subseteq E_{\mathscr{E}} : e\sharp_{\mathscr{E}}f(i \in \mathbf{N}, 1 \leq i \leq m)$.

If $A_i\sharp^{xs}B_i$, then any nonempty subset of A_i and any nonempty subset of B_i are also in extended strong conflict.

For any event set D, let $\widehat{D}$ denote any nonempty subset of D ($\emptyset \neq \widehat{D} \subseteq D$); correspondingly, $\widehat{\mathscr{I}^{i}_{\sharp_{\mathscr{E}}}} : \widehat{A_i}\sharp^{xs}\widehat{B_i}$ denotes a conflict subpattern of $\mathscr{I}^{i}_{\sharp_{\mathscr{E}}}$ ($i \in \mathbf{N}, 1 \leq i \leq m$).

Formally, $\mathscr{I}' : A'\sharp^{xs}B'$ is called a *conflict subpattern* of $\mathscr{I} : A\sharp^{xs}B$ if and only if $(\emptyset \neq A' \subseteq A) \wedge (\emptyset \neq B' \subseteq B)$, denoted by $\mathscr{I}'\subseteq_{\text{pattern}}\mathscr{I}$ (or $\mathscr{I}\supseteq_{\text{pattern}}\mathscr{I}'$). Otherwise, $\mathscr{I}'\nsubseteq_{\text{pattern}}\mathscr{I}$ (or $\mathscr{I}\nsupseteq_{\text{pattern}}\mathscr{I}'$).

Let $P_{\mathscr{E}} = \{\mathscr{I}^{i}_{\sharp_{\mathscr{E}}} : A_i\sharp^{xs}B_i \mid i \in \mathbf{N}, 1 \leq i \leq m\}$ denote the maximal conflict pattern set of an event structure $\mathscr{E}$.

For any prime event structure, it is a certainty that we can get its maximal conflict-free substructures by some kind of conflict-free partition operation according to its conflict relation characteristics: *maximal conflict patterns*. Thus, we have the following theorem for partition.

Theorem 9. *For any prime event structure, its maximal conflict-free partition exists and the partition result is unique.*

Proof. (1) Existence. The proof is constructive.

If there is no conflict in $E_{\mathscr{E}}$, then $E_{\mathscr{E}}$ itself is the maximal conflict-free event subset of $\mathscr{E}$. Otherwise, for any nonempty event subset E ($E \subseteq E_{\mathscr{E}} \wedge E \neq \emptyset$), and there exists such maximal conflict pattern $\mathscr{I}^{i}_{\sharp_{\mathscr{E}}} : A_i\sharp^{xs}B_i$ ($i \in \mathbf{N}, 1 \leq i \leq m$) that $(\widehat{A_i} \cup \widehat{B_i}) \subseteq E$.

In order to make a subset E' of E ($E' \subseteq E$) become conflict-free with respect to the conflict relation: $A_i\sharp^{xs}B_i$, that is, eliminate this conflict relation from its all subsets, we have known that if $\widehat{A_i} \subseteq E'$ (or $\widehat{B_i} \subseteq E'$), then there should be $B_i \nsubseteq E'$(or $A_i \nsubseteq E'$); otherwise, the $\mathscr{I}^{i}_{\sharp_{\mathscr{E}}}$ conflict pattern will still exist in its subsets.

By greedy policy, let A'_i and B'_i be both maximal inclusion subsets with respect to E calculated by $A'_i = \mathrm{maxincl}(A_i, E)$ and $B'_i = \mathrm{maxincl}(B_i, E)$, respectively. This means the current event set E will be partitioned into two parts by this maximal conflict subpattern: one part is $(E - A'_i)$, and the other is $(E - B'_i)$. Certainly, there exists no $\mathscr{I}^{i}_{\sharp_{\mathscr{E}}}$ conflict relation between $(E - A'_i)$ and $(E - B'_i)$ any more. If there does not exist any conflict in $(E - B'_i)$ (or $(E - A'_i)$), then $(E - B'_i)$ (or $(E - A'_i)$) is one conflict-free event subset of $E_{\mathscr{E}}$.

Otherwise, apply the next maximal conflict pattern $\mathcal{F}^{i+1}_{\sharp_{\mathscr{E}}}$ $(i \in \mathbf{N}, 1 \leq i \leq m)$ to all the previously obtained event subsets in the same manner. This partition process is continued until no conflict exists.

As we know, if each pattern of the maximal conflict patterns set has been applied just once by the above manner, then any consequent subset will be conflict-free and the partition process will stop. Meanwhile, there are 2^m conflict-free subsets at most.

Because intersection of x_i and y_i $(x, y \in \{A, B\},\ i \neq j, 1 \leq i, j \leq m)$ can be nonempty, thus the partition tree is not yet a full binary tree and set inclusion among these solution nodes is allowed. If some subsets are included by others, then they will be removed until every result subset cannot be included by others. It is not difficult to verify that every consequent subset is maximal and conflict-free. Exploiting these expanded fully conflict patterns to partition the event set $E_{\mathscr{E}}$ step by step, we will eventually get all maximal conflict-free event subsets. That is, there exists a practical algorithm to implement the partition operation. Without loss of generality, let $\oslash_A$ denote such partition for the time being.

(2) Uniqueness. Assume we have $mcfsetN_A(\mathscr{E})$ distinct maximal conflict-free event subsets in total by partition $\oslash_A$. These subsets form a set of *mcfsets*, denoted by $\mathbf{R}_A = \{A_1, A_2, \ldots, A_n \mid n = mcfsetN_A(\mathscr{E})\}$.

We might as well assume there is another partition $\oslash_B$ that generates the result set $\mathbf{R}_B = \{B_1, B_2, \ldots, B_m \mid m = mcfsetN_B(\mathscr{E})\}$ which is also a set of *mcfsets*.

Consider any element of $\mathbf{R}_B$; let B_i $(1 \leq i \leq m, i \in \mathbf{N})$ denote it. The relationship between an element A_j $(1 \leq j, k \leq mcfsetN_A(\mathscr{E}), j, k \in \mathbf{N})$ in $\mathbf{R}_A$ and B_i satisfies the following.

(1) $\exists A_j \in \mathbf{R}_A : A_j \subset B_i \subseteq E_{\mathscr{E}}$.

Since $\forall A_k \in \mathbf{R}_A$, $k \neq j$: $A_j \sharp^w A_k$, then $\forall A_k \in \mathbf{R}_A$, $k \neq j$: $B_i \sharp^w A_k$. We have known that $A_j \in \mathbf{R}_A$ is maximal, and now event subset B_i is also a subset of $E_{\mathscr{E}}$ and is in weak conflict with other event subsets except A_i. Moreover, B_i includes A_i. This case leads to a contradiction.

(2) $\exists A_j \in \mathbf{R}_A : B_i \subset A_j \subseteq E_{\mathscr{E}}$.

The proof is similar to the above case (1). This case also leads to a contradiction.

(3) $\forall A_j \in \mathbf{R}_A : A_j \neq B_i$.

(3.1) $\forall A_j \in \mathbf{R}_A : A_j \sharp^w B_i$.

Since B_i is also a subset of $E_{\mathscr{E}}$, thus, $\mathbf{R}'_A = \{A_1, A_2, \ldots, A_n, B_i \mid n = mcfsetN_A(\mathscr{E})\}$ is a valid set of *mcfsets*. There are $n + 1$ subsets in this partition $\oslash_A$. This is in contradiction with that there are n $(n = mcfsetN_A(\mathscr{E}))$ subsets in $\mathbf{R}_A$.

(3.2) $\exists A_j \in \mathbf{R}_A : \neg(A_j \sharp^w B_i)$.

Since $\forall A_k \in \mathbf{R}_A$, $k \neq j$: $A_j \sharp^w A_k$, then $\forall A_k \in \mathbf{R}_A$, $k \neq j$: $(A_j \cup B_i) \sharp^w A_k$; that is, $\mathbf{R}''_A = \{A_1, A_2, \ldots, \{A_j \cup B_i\}, \ldots, A_n, \mid n = mcfsetN_A(\mathscr{E})\}$ is a valid set of *mcfsets*. A_j $(A_j \in \mathbf{R}_A)$ is maximal; moreover, $A_j \subset (A_j \cup B_i) \in \mathbf{R}''_A$ is maximal too. This leads to a contradiction.

(3.3) $\forall A_j \in \mathbf{R}_A : \neg(A_j \sharp^w B_i)$.

The proof is similar to the above case (3.2). This case also leads to a contradiction.

Therefore, we are forced to have only $\exists A_j \in \mathbf{R}_A : A_j = B_i$; that is, any element in $\mathbf{R}_B$ is also an element in $\mathbf{R}_A$; we get $\mathbf{R}_B \subseteq \mathbf{R}_A$; in the same manner, we will get $\mathbf{R}_A \subseteq \mathbf{R}_B$. Thus, we have $\mathbf{R}_B = \mathbf{R}_A$.

This establishes the uniqueness and also implies the partition result is independent of partition order or conflict pattern.

Therefore, we have the conclusion.

Assume $\mathscr{E} \in \mathbb{E}$ has $mcfsetN(\mathscr{E}) \in \mathbf{N}$ *mcfESs* in total. Here, let $N_{\mathscr{E}} = mcfsetN(\mathscr{E})$ ($N_{\mathscr{E}}$, for short) denote total amount of *mcfESs*, $\mathscr{E}^{\max}_i$ $(i \in N, 1 \leq i \leq N_{\mathscr{E}})$ denote the *i*th maximal conflict-free event substructure, and $E_{\mathscr{E}^{\max}_i}$ denote the event set of $\mathscr{E}^{\max}_i$.

Then the result set can be represented as $\mathscr{B}^{mcfES}(\mathscr{E}) = \{\mathscr{E}^{\max}_i \mid i \in \mathbf{N}, 1 \leq i \leq N_{\mathscr{E}}\}$.

In fact, every $\mathscr{E}^{\max}_i$ $(i \in, 1 \leq i \leq mcfsetN(\mathscr{E}))$ of the original prime event structure represents a specific possible execution choice in a system run. We might as well let *cfp* denote such an operator. Then, we have the following definition of this partition operator.

Definition 10 (conflict-free partition). An operator *cfp* is called *conflict-free partition* operator for $\mathscr{E} \in \mathbb{E}$ if and only if $\mathscr{B}^{mcfES}(\mathscr{E}) = cfp(\mathscr{E})$.

According to our previous discussion, we have C-like pseudocode descriptions: Algorithm 1 for *cfp*.

4.2. Family of Configurations. In general, the behavior of an event structure is described by its configurations which are sets of events with certain properties. In other words, a configuration is a set of events that have happened during a specific run of the event structure.

We will review the basic definition of configuration in the following section. More detailed information can be found in [26].

Definition 11 (configuration). Let X be a subset of $X \subseteq E_{\mathscr{E}}$ of a prime event structure $\mathscr{E} \in \mathbb{E}$; then X is called a *configuration* of $\mathscr{E}$ if and only if

(1) X is left-closed if and only if $\forall c, d \in E, d \in X \wedge c \leq d \Rightarrow c \in X$.

(2) X is conflict-free if and only if $\forall e, f \in X : \neg(e \sharp_{\mathscr{E}} f)$.

A configuration can also be viewed as a global state where all events in the configuration have occurred. The configuration of the event structure should be conflict-free because conflicting events can never happen in a system run. In addition, all casual predecessors of an event in

```
Input: a prime event structure: $\mathscr{E}$;
Output: the set of $mcfESs$: $\mathscr{B}^{mcfES}(\mathscr{E})$;
BEGIN
(1)  $C = \emptyset; int\ i = 1; int\ n = 0; R = \emptyset$;
(2)  $SQ = EnterQueue(E_{\mathscr{E}}, 1)$;
     /* Initialize the Queue */
(3)  if ($IsConflictFree(E_{\mathscr{E}})$){
(5)      $C = \cup\{E\}$;
(6)      Goto BUILDES;
(7)  } /* end if; */
     /* Expand and merge each conflict pair and build maximal conflict patterns:
     $\{\mathscr{I}^{i}_{\#_{\mathscr{E}}} : A_i \#^{xs} B_i \mid i \in, 1 \le i \le m\}$; */
(8)  for ($i = 1; i \le m; i{+}{+}$) { /* do */
(9)      Select a partition pattern: $\mathscr{I}^{i}_{\#_{\mathscr{E}}} : A_i \#^{xs} B_i$;
(10) /* Current level is $i$, apply $\mathscr{I}^{i}_{\#_{\mathscr{E}}}$, otherwise, skip while loop to the next one: $i{+}{+}$; */
(11)     while (!$EmptyQueue(SQ)$ &&
         $i == VarQueue(SQ, \mathbf{LEVEL})$) {
     /* Get the head of the Queue: event set */
(12)         $E = OutQueue(SQ, \mathbf{EVENT})$;
(13)         $A_i' = maxincl(A_i, E)$;
(14)         $B_i' = maxincl(B_i, E)$;
(15)         $E_1 = (E - A_i')$;
(16)         $E_2 = (E - B_i')$;
     /* $E_1$ is a conflict-free subset; */
(17)         if ($IsConflictFree(E_1)$) {
(18)             $C = C \cup \{E_1\}$;
     /* Remove these elements included in others; */
(19)             $RemoveIncludedElement(C)$;
(20)         }else{
     /* Continue next partition by the conflict pattern: $\mathscr{I}^{i+1}_{\#_{\mathscr{E}}}$; */
(21)             $SQ = EnterQueue(E_1, i + 1)$;
(22)         } /* end if */
(23)         if ($IsConflictFree(E_2)$) {
(24)             $C = C \cup \{E_2\}$;
(25)             $RemoveIncludedElement(C)$;
(26)         }else{
(27)             $SQ = EnterQueue(E_2, i + 1)$;
(28)         } /* end if */
(29)     } /* end while */
(30) } /* end for */
(31) BUILDES: /* Build the sub-structure: $\mathscr{F} = (E_{\mathscr{F}}, \preceq_{\mathscr{F}}, \#_{\mathscr{F}}, l_{\mathscr{F}})$ */
(32) while ($\neg IsEmpty(C)$) {
(33)     Select a conflict-free event subset $A_{!\#}$ from $C$;
(34)     $E_{\mathscr{F}} = A_{!\#}$;
(35)     $\#_{\mathscr{F}} = \emptyset$;
(36)     $\preceq_{\mathscr{F}} = \preceq_{\mathscr{E}} \cap (A_{!\#} \times A_{!\#})$;
(37)     $l_{\mathscr{F}} = l|_{A_{!\#}}$;
(38)     $C = C - \{A_{!\#}\}$;
(39)     $R = R \cup \mathscr{F}; n{+}{+}$;
(40) } /* end while */
(41) $mcfsetN(\mathscr{E}) = n$;
(42) return $R$;
END;
```

Algorithm 1: Conflict-free partition: *cfp*.

a configuration should be contained in this configuration too; that is, configuration should be downwards closed; otherwise this event could not have happened at all.

That is, a subset X is a (finite) configuration of $\mathscr{E}$ if and only if it is finite, left-closed, and conflict-free.

The semantics of a prime event structure is defined as the family of its configurations ordered by set inclusion. Let $ConfF(\mathscr{E})$ denote the family of all configurations of event structure $\mathscr{E}$, which forms an ordered set (called prime algebraic coherent partial order; see [16]) by inclusion; that is, $(ConfF(\mathscr{E}), \subseteq)$ is partial order.

Definition 12. A configuration $X \in \mathrm{Conf}F(\mathscr{E})$ is called *complete* or (successfully) *terminated* if and only if $\forall d \in E : d \notin X \Rightarrow \exists e \in X : e \sharp d$. A configuration $X \in \mathrm{Conf}F(\mathscr{E})$ is called *maximal* if and only if $\forall Y \in \mathrm{Conf}F(\mathscr{E}) : X \not\subset Y$.

For any prime event structure $\mathscr{E}$, a configuration of $\mathscr{E}$ is maximal if and only if it is complete. Obviously, for any maximal configuration of a prime event structure, there exists a corresponding maximal conflict-free substructure set. An empty or initial configuration, denoted by $\emptyset_{\mathrm{ConfF}} \in ConfF(\mathscr{E})$, represents the initial state in which there is no event happened.

In general, initial configuration and complete configuration are also called trivial configurations, while others are called nontrivial configurations.

Similarly, we have the following configuration definition for conflict-free event structure.

Definition 13 (configuration of $cfES$). Let $\mathscr{F} = (E_{\mathscr{F}}, \preceq_{\mathscr{F}}, \emptyset, l_{\mathscr{F}})$ be a $cfES$ and let Z be a subset of $E_{\mathscr{F}}$ ($Z \subseteq E_{\mathscr{F}}$); then Z is called a configuration of $\mathscr{F}$ if and only if Z is left-closed; that is, $\forall c,\ d \in E_{\mathscr{F}},\ d \in Z \wedge c \preceq d \Rightarrow c \in Z$.

Since $\mathscr{F} \in \mathbb{F}$, its event subset is evidently conflict-free.

Let $cf\mathrm{Conf}F(\mathscr{F})$ denote the family of all configurations of conflict-free event structure $\mathscr{F}$. Clearly, when $\mathscr{F}$ is the ith $mcfES$: $\mathscr{E}_i^{\max}$ ($i \in \mathbf{N}, 1 \le i \le N_{\mathscr{E}}$) of prime event structure $\mathscr{E} \in \mathbb{E}$, its family of all configurations is denoted by $mcf\mathrm{Conf}F(\mathscr{E}_i^{\max})$.

Definition 14 (subfamily of configurations). Let $\mathscr{F} \in$ be a $cfES$ and let Ω ($\emptyset \neq \Omega \subseteq E_{\mathscr{F}}$) be a nonempty event subset; a subfamily of configurations of $\mathscr{F}$ with respect to event subset Ω is the family of configurations of its event substructure $\mathscr{F}|_{\Omega} = (\Omega, \le_{\mathscr{F}} \cap (\Omega \times \Omega), \emptyset, l_{\Omega})$ restricted by event subset Ω; that is, $cf\mathrm{subConf}F(\mathscr{F}, \Omega) \triangleq cf\mathrm{Conf}F((\Omega, \le_{\mathscr{F}} \cap (\Omega \times \Omega), \emptyset, l_{\Omega}))$.

Clearly, for any $mcfES\mathscr{E}_i^{\max}$ ($i \in \mathbf{N}, 1 \le i \le N_{\mathscr{E}}$) of event structure $\mathscr{E} \in \mathbb{E}$, its subfamily of configurations with respect to event subset Ω ($\emptyset \neq \Omega \subseteq E_{\mathscr{E},i}^{\max}$) is denoted by $mcf\mathrm{subConf}F(\mathscr{E}_i^{\max}, \Omega) \triangleq mcf\mathrm{Conf}F((\Omega, \le_{\mathscr{E}_i^{\max}} \cap (\Omega \times \Omega), \emptyset, l_{\Omega}))$ for convenience.

Lemma 15. *The relation between the family of configurations of a prime event structure and that of its mcfESs can be described by* $ConfF(\mathscr{E}) = \bigcup_{1 \le i \le N_{\mathscr{E}}} (mcf\,ConfF(\mathscr{E}_i^{\max}))$.

Proof. To prove the result of this lemma, we will show that

$$\bigcup_{1 \le i \le N_{\mathscr{E}}} (mcf\mathrm{Conf}F(\mathscr{E}_i^{\max})) \subseteq \mathrm{Conf}F(\mathscr{E}),$$
$$\mathrm{Conf}F(\mathscr{E}) \subseteq \bigcup_{1 \le i \le N_{\mathscr{E}}} (mcf\mathrm{Conf}F(\mathscr{E}_i^{\max})) \tag{2}$$

both hold.

(1) "$\subseteq$". For any configuration $X \in \mathrm{Conf}F(\mathscr{E})$, since X is a configuration, by definition, X should be conflict-free. Thus X should be the subset of one of the maximal configurations. Otherwise, if X is greater than any maximal configuration, then X must contain mutual conflicting events; that is impossible.

Therefore, we have that there must exist a maximal configuration which contains X. Such a maximal configuration corresponds to a maximal conflict-free event subset: $E_{\mathscr{E}_i^{\max}} \in$ ($i \in \mathbf{N}, 1 \le i \le N_{\mathscr{E}}$); that is, X must be the element of $mcf\mathrm{Conf}F(\mathscr{E}_i^{\max})$; that is, $X \in mcf\mathrm{Conf}F(\mathscr{E}_i^{\max})$. We have $\mathrm{Conf}F(\mathscr{E}) \subseteq \bigcup_{1 \le i \le N_{\mathscr{E}}} (mcf\mathrm{Conf}F(\mathscr{E}_i^{\max}))$.

(2) "$\supseteq$". For any configuration $X \in mcf\mathrm{Conf}F(\mathscr{E}_i^{\max})$ ($i \in \mathbf{N}, 1 \le i \le N_{\mathscr{E}}$), of course, $X \in \bigcup_{1 \le i \le N_{\mathscr{E}}} (mcf\mathrm{Conf}F(\mathscr{E}_i^{\max}))$; this implies $X \subseteq E_{\mathscr{E}_i^{\max}}$ and $E_{\mathscr{E}_i^{\max}} \subseteq E_{\mathscr{E}}$; therefore, we get $X \subseteq E_{\mathscr{E}}$. Since X is a configuration, it is also a configuration of $\mathscr{E}$; that is, $X \in ConfF(\mathscr{E})$.

We have $\cup_{1 \le i \le N_{\mathscr{E}}} (mcf\mathrm{Conf}F(\mathscr{E}_i^{\max})) \subseteq \mathrm{Conf}F(\mathscr{E})$.

Therefore, from (1) and (2), we have the result.

4.3. Domains of Configurations. In this section, we will discuss the concept of domain from the point of view that computation states are taken as such subsets and progress in a computation is measured by the occurrence of more events.

Firstly, we will recall some related conceptions regarding domain [16, 27]. Then, some important facts will be discussed.

Definition 16 (least upper bound). Let $\mathbf{D} = (D, \sqsubseteq)$ be a partial order; an element $d \in D$ is called *least upper bound* of subset X ($X \subseteq D$), denoted by $d = \sqcup X$, if and only if $(\forall x \in X : x \sqsubseteq d) \wedge (\forall d' \in D : (\forall x \in X : x \sqsubseteq d') \Rightarrow d \sqsubseteq d')$.

Definition 17 (coherent). Let $\mathbf{D} = (D, \sqsubseteq)$ be a partial order; two elements $x, y \in D$ are called *consistent* (denoted by $x \uparrow y$) if and only if $\exists z \in D : x \sqsubseteq z \wedge y \sqsubseteq z$; a subset $X \subseteq D$ is *pairwise consistent* if and only if any two of its element have an upper bound in D; that is, $\forall x, y \in X : x \uparrow y$; $(D, \sqsubseteq)$ is called *coherent* if and only if every pairwise consistent subset X ($X \subseteq D$) has a least upper bound $\sqcup X$.

The *consistency* relation of x and y is denoted by $x \uparrow y$; conversely, *inconsistency* is denoted by $x \not\uparrow y$.

Definition 18 (complete prime). A partial order $\mathbf{D} = (D, \sqsubseteq)$; an element is a *complete prime* if and only if for every finite subset $X \subseteq D$, if $\sqcup X$ exists and $p \subseteq \sqcup X$ then there exists an $x \in X$ such that $p \sqsubseteq x$ (i.e., $p \sqsubseteq \sqcup X \Rightarrow \exists x \in X. p \sqsubseteq x$).

Let $P(D)$ denote the set of complete prime of $(D, \sqsubseteq)$.

Definition 19 (prime algebraic). A partial order $\mathbf{D} = (D, \sqsubseteq)$ is called *finitary* if and only if $\forall p, d \in P(D) : \{d \in D \mid d \sqsubseteq p\}$ is finite. $(D, \sqsubseteq)$ is called *prime algebraic* if and only if $P(D)$ is countable and $\forall d \in D : d = \sqcup\{p \in P(D) \mid p \sqsubseteq d\}$.

Namely, $\mathbf{D}$ is called *prime algebraic* if and only if, for every element $d \in D$, $\sqcup\mathbf{D}_d$ exists (define $\sqcup\mathbf{D}_d = \{p \sqsubseteq d \mid p \text{ is a complete prime}\}$), and $d = \sqcup\mathbf{D}_d$.

Definition 20 (domain). A coherent, prime algebraic, and finitary partial order is called a *Scott domain* (or simply a *domain*).

Definition 21. Let $\mathbf{D} = (D, \sqsubseteq)$ be a *coherent, finitary prime algebraic domain.* Define $\mathscr{P}_r[D] = (P(D), \preceq, \sharp)$, where $P(D)$ consists of the complete primes of $\mathbf{D}$:

(1) $\forall p, p' \in P(D) : p \preceq p' \Leftrightarrow p \sqsubseteq p'$;

(2) $\forall p, p' \in P(D) : p \sharp p' \Leftrightarrow p \not\uparrow p'$.

Definition 22. Let $\mathbf{D} = (D, \sqsubseteq)$ be a *prime algebraic complete lattice.* Define $\mathscr{P}_r[D] = (P(D), \preceq)$, where $P(D)$ consists of the complete primes of $\mathbf{D}$, $\forall p, p' \in P(D) : p \preceq p' \Leftrightarrow p \sqsubseteq p'$.

Theorem 23. *Let $\mathscr{E} = (E_{\mathscr{E}}, \preceq_{\mathscr{E}}, \sharp_{\mathscr{E}}, l_{\mathscr{E}}) \in \mathbb{E}$; then $(ConfF(\mathscr{E}), \subseteq)$ is a finitary coherent prime algebraic domain; the complete primes are the set $\{a \in E_{\mathscr{E}} \mid a \preceq e, e \in E_{\mathscr{E}}\}$ (see [25]).*

Theorem 24. *Let $(D, \sqsubseteq)$ be a finitary coherent prime algebraic domain. Then, $\mathscr{P}_r[D] = (P(D), \preceq, \sharp)$ is a prime event structure, with $\varphi : (D, \sqsubseteq) \cong (ConfF(\mathscr{P}_r[D]), \subseteq)$ giving an isomorphism of partial orders where $\varphi(d) = \{p \sqsubseteq d \mid p \text{ is a complete prime}\}$ with inverse $\lambda : ConfF(\mathscr{P}_r[D]) \to (D, \sqsubseteq)$ given by $\lambda(x) = \sqcup x$ (see [16]).*

Evidently, event structures and coherent, finitary prime algebraic domains are equivalent; one can be used to represent the other.

The following theorem describes the important property of family of configurations of a prime event structure.

Theorem 25. *For any nonempty mcfES: $\mathscr{E}_i^{\max}$ ($i \in \mathbf{N}, 1 \leq i \leq N_{\mathscr{E}}$) of event structure $\mathscr{E} \in \mathbb{E}$, its family of configurations $(mcfConfF(\mathscr{E}_i^{\max}), \subseteq)$ is prime algebraic complete lattice. Its complete primes are those elements of the form $\{a \in E_{\mathscr{E}_i^{\max}} \mid a \preceq e, e \in E_{\mathscr{E}_i^{\max}}\}$.*

Proof. The proof is straightforward.

Thus prime event structure and finitary coherent prime algebraic domain are equivalent; this implies that there is a one-to-one correspondence between a prime event structure and its family of configurations; one can be used to represent the other.

4.4. Weak Choice Composition. Theorem 23 describes an important property between the domains of configurations of prime event structures and the prime event structures themselves.

We can obtain a full set of *mcfESs* from a prime event structure by applying *cfp* operator over it. Conversely, given a full set of *mcfESs* of an event structure, we can certainly recover the original event structure that generates this set of *mcfESs* by some kind of composition operation.

Further, for any weak conflict set, we give the constraint conditions, under which this weak conflict set can be composed together and form a prime event structure that can generate this set by conflict-free partition operation.

The following theorem discusses the constraint conditions for composition.

Theorem 26 (necessary and sufficient condition for composition). *For any weak conflict set $\mathscr{WF}_n^{cfw} = \{\mathscr{F}_i \in \mathbb{F} \mid i, n \in \mathbf{N}, 1 \leq i \leq n\}$, if it satisfies the following conditions: (1) and (2), then there exists a unique prime event structure $\mathscr{E} \in \mathbb{E}_{prime}$ that can generate this set by cfp partition operation; that is, $\mathscr{WF}_n^{cfw} = cfp(\mathscr{E})$.*

(1) $\forall \mathscr{F}_i, \mathscr{F}_j \in \mathscr{WF}_n^{cfw} : \Omega = (E_{\mathscr{F}_i} \cap E_{\mathscr{F}_j}) \wedge (\Omega \neq \emptyset) \Rightarrow cfsubConfF(\mathscr{F}_i, \Omega) = cfsubConfF(\mathscr{F}_j, \Omega)$ $(i, j \in N, i \neq j, 1 \leq i \leq n)$.

(2) $(\bigcup_{i=1}^{n} cfConfF(\mathscr{F}_i), \subseteq)$ *is a finitary coherent prime algebraic domain.*

Proof. On one hand, the intersection of event sets of any two *cfESs* is nonempty meaning that common events have happened from both event structures. By definition, if these events represent common global states in runs of a system described by the same prime event structure with multiple choices, they should behave identically. That is, their configurations with respect to the intersection of event set should be identical.

In addition, from Theorems 23 and 24, the family of configurations of a prime event structure ordered by set inclusion should be a finitary coherent prime algebraic domain.

Thus, we have the necessary condition for composition.

On the other hand, from Theorems 23 and 24, we have that there is a one-to-one correspondence between a prime event structure and its family of configurations. Given a valid family of configurations for prime event structure, then there should exist a corresponding prime event structure.

For any weak conflict set: $\mathscr{WF}_n^{cfw} = \{\mathscr{F}_i \in \mathbb{F} \mid i, n \in, 1 \leq i \leq n\}$, if all $cfConfF(\mathscr{F}_i)$ by joining can form a valid family of configurations for a prime event structure, that is, $\bigcup_{1 \leq i \leq n} cfConfF(\mathscr{F}_i)$ forms a ordered by set inclusion, then there should exist such a unique prime event structure $\mathscr{E}$ that $ConfF(\mathscr{E}) = \bigcup_{1 \leq i \leq n} cfConfF(\mathscr{F}_i)$.

Therefore, we get the necessary and sufficient condition for composition.

Obviously, the set $\mathscr{B}^{mcfES}(\mathscr{E})$ of a prime event structure satisfies the above condition. Clearly, this implies that there must exists a composition operation which can construct the target event structure $\mathscr{E}$ from a weak conflict set that satisfies the constraint conditions. We may as well let *wcc* denote the operator. Thus, we have the following definition.

Definition 27 (weak choice composition (*wcc* operator)). Let $\mathscr{W}\mathscr{F}_n^{cfw}$ be a weak conflict set, which satisfies necessary and sufficient conditions for composition; an operator *wcc* is called *weak choice composition* operator if and only if the result event structure $\mathscr{R} = wcc(\mathscr{W}\mathscr{F}_n^{cfw})$ and $\mathscr{R}$ satisfies the following:

(1) $\mathrm{Conf}F(R) = (\bigcup_{i=1}^{n} cf\mathrm{Conf}F(\mathscr{F}_i), \subseteq)$;

(2) $\mathscr{W}\mathscr{F}_n^{cfw} = cfp(R)$.

The following theorem states that the operator *wcc* and *cfp* are mutually inverse for a prime event structure.

Theorem 28. *For any $\mathscr{E} \in \mathbb{E}$, $\mathscr{E} = wcc(cfp(\mathscr{E}))$ holds.*

Proof. The proof is straightforward.

Obviously, it is not difficult to derive an algorithm for weak conflict composition operator from Definition 27 and Theorem 28.

5. Slicing Reduction

In this section, we will discuss slicing reduction technique for partial order trace or prime event structure. Slicing is often taken as an effective abstract technique to combat the state explosion problem. A slicing algorithm for event structure with respect to predicates in a subset of temporal logic formulas is studied. Specially, we focus on statically analyzing rather than online detecting over event structure model.

First of all, we will retrospect the classical notion of computation slicing for partial order traces. Then, we will extend the idea from partial order traces to prime event structures with conflict relations. Additionally, all related definitions and theorems [18, 19, 28] for our theory will be discussed.

5.1. Partial Order Trace Slicing. Computation slicing was introduced in [7] as an abstraction technique for analyzing partial order traces of distributed programs or distributed computations.

Generally, for classical program slicing, programs are sliced with respect to a slicing criterion that is an interested point for analyzing. In static program slicing, for example, "a program line number" can be taken as a valid slicing criterion. Thus, in order to compute a slice, we need to firstly define the slicing criterion.

Intuitively, a slice of a trace with respect to a temporal logic specification or a predicate (slicing criterion) φ is a subtrace that contains all the states of the trace that satisfy φ. A slice contains all the states that satisfy φ such that it can be computed efficiently and is often much smaller than the original model.

We can use directed graphs to model partial order (execution) traces (POTs, for short) as well as slices. Thus, a notion named *graph ideal* (or *order ideal*) of directed graph [29] is introduced to specify partial order traces and slices pictorially. Formally, its definition is given as follows.

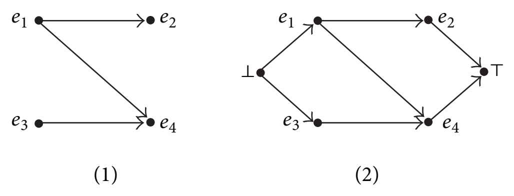


Figure 1: Partial order trace and its directed graph representation.

Definition 29 (order ideal). Given a poset $(X, \leq)$, ($\leq$ denotes an order relation) a subset S of X is an *order ideal* if it satisfies $\forall x, y : x, y \in X : (x \in S) \wedge (y \leq x) \Rightarrow (y \in S)$.

Definition 30 (graph ideal). Given a directed graph $G = (V, \Gamma)$, let $V(G)$ and $\Gamma(G)$ denote the set of vertices with event labels and directed edges, respectively. A subgraph H of G is a *graph ideal* if it satisfies $\forall u, v \in V(G), H \subseteq V(G), v \in H \wedge (u, v) \in \Gamma(G) \Rightarrow u \in H$.

It is more convenient to use directed graphs to represent partially ordered sets and prime event structures for slicing computation. It satisfies the following.

(1) For any event e and f of $\mathscr{E}$, if $e \preceq f$, then there is directed edge from the vertex v_e labelled with e to the vertex v_f labelled with f.

(2) For any event e and f of $\mathscr{E}$, if $e \sharp f$, then there is dash line between the vertex v_e labelled with e to the vertex v_f labelled with f.

For example, as shown in Figure 1, a partial order trace or a *mcfES* is demonstrated pictorially. The corresponding event structure for Figure 1 is as follows.

(i) $\mathscr{E} = (E_{\mathscr{E}}, \preceq_{\mathscr{E}}, \sharp_{\mathscr{E}}, l_{\mathscr{E}}), (Act = \{a, b, c, d\})$.

(1) $E_{\mathscr{E}} = \{e_1, e_2, e_3, e_4\}$.
(2) $\preceq_{\mathscr{E}} = \{e_1 \preceq e_2, e_1 \preceq e_4, e_3 \preceq e_4\}$.
(3) $l(e_1) = a,\ l(e_2) = b,\ l(e_3) = c,\ l(e_4) = d$.
(4) $\sharp_{\mathscr{E}} = \emptyset$.

In addition, when attempting to construct the graph representation of *mcfES*, as Figure 1 shows, two specific vertexes $\top$ and $\perp$ will be added as initial state and terminal state corresponding to initial configuration and maximal configuration, respectively.

A subset of elements forms an order ideal if whenever an element is contained in the subset then all its preceding elements are also contained in the subset. Intuitively, order ideals or left-closed subsets can be graphically represented by graph ideals. Generally, independency relation will not be represented explicitly. It is not difficult to have that partial order trace is only a special case of prime event structure with no conflict relations. Here, graph ideal is a notion equivalent to the configuration of an event structure. Empty set and the set of all vertices are called trivial ideal. Similarly, initial configuration and complete configuration are also called trivial configurations.

Definition 31 (predicate on configuration). Intuitively, a logic formula or predicate is a Boolean-valued function defined on the set of configurations: $\varphi : \mathrm{Conf}F(\mathscr{E}) \to \{0, 1\}$. It actually represents a subset of configurations in which the Boolean function evaluates to 1.

The predicate detection problem is to decide whether the initial configuration of an event structure satisfies a predicate. More formally, we have the following definition.

Definition 32 (predicate detection). For any prime event structure $\mathscr{E}$ and any predicate φ, predicate detection is to decide whether $\mathrm{Conf}F(\mathscr{E}), \{\perp\} \models \varphi$ holds or not.

Predicates are used to specify system behaviors and properties such as safety and liveness. Properties expressed by a CTL (computational tree logic, introduced in [30]) formula are beyond the scope of this paper. For evaluating the value of a predicate efficiently, various predicate classes [28] such as conjunctive, stable, observer-independent, linear, relational, and nontemporal regular [7] predicates have been defined.

Generally, predicate on configurations will act as the slicing criterion for POTs slicing.

Definition 33 (slice of *mcfES*(POTs)). A *slice* of a *mcfES*(POTs): $\mathscr{E}_i^{\max}$ $(i \in \mathbf{N}, 1 \le i \le N_{\mathscr{E}})$ of prime event structure $\mathscr{E}$ with respect to a formula φ, denoted by $mcfES\mathrm{slice}(\mathscr{E}_i^{\max}, \varphi)$, is such an event structure that satisfies the following.

(i) Its family of configurations contains all the configurations that satisfy φ.

(ii) Its family of configurations has the least number of configurations and still forms a sublattice.

This formal definition is derived from computation slice notion [7] given by Garg and Mittal. Meanwhile, existence and uniqueness of the *mcfES* slice have also been discussed; that is, the following theorem holds.

Theorem 34. *For any $\mathscr{E}_i^{\max}$ $(i \in N, 1 \le i \le N_{\mathscr{E}})$ of a prime event structure and any predicate φ, the slice of with respect to predicate φ, that is, $mcfESslice(\mathscr{E}_i^{\max}, \varphi)$ exists and is unique.*

Proof. The proof is straightforward; see [8, 20, 31].

In general, the family of configurations for a *mcfES* forms a distributed lattice, and its slice with respect to a predicate is a sublattice. Sometimes a slice may contain those configurations that do not satisfy the predicate for completing sublattice.

In the next section, we will discuss the slicing definition and model for prime event structure.

5.2. Sliced Model over Event Structure. Generally, predicate on configurations acts as the slicing criterion for prime event structure slicing. Temporal regular predicate, such as a regular subset of CTL called RCTL [7, 8, 29], which contains four temporal operators EF, AG, EG, and EX[j], and nontemporal regular predicates both can also be taken as the slicing criterions.

Compared with the definition of slice of *mcfES*, we have a similar case for prime event structure.

Definition 35 (slice of prime event structure). A slice of a prime event structure with respect to a formula φ, denoted by $SliceES(\mathscr{E}, \varphi)$, is such an event structure that satisfies the following.

(i) Its family of configurations contains all the configurations that satisfy φ.

(ii) Its family of configurations has the least number of configurations.

Generally, a slice may contain configurations that do not satisfy the given predicate. The slice of an event structure with respect to a predicate is called *lean* [32] if every configuration of the slice satisfies the predicate.

Theorem 36. *For any $\mathscr{E} \in \mathbb{E}$ and any predicate φ, $SliceES(\mathscr{E}, \varphi)$ exists and is unique, and $SliceES(\mathscr{E}, \varphi) = wcc(mcfESslice(cfp(\mathscr{E}), \varphi))$ holds.*

Proof. (1) Existence and Uniqueness. From Theorem 34, we have that, for any $\mathscr{E}_i^{\max}$ $(i \in N, 1 \le i \le N_{\mathscr{E}})$ of a prime event structure $\mathscr{E}$ and any predicate φ, its slice with respect to predicate φ exists and is unique.

For any *mcfES*, the family of configuration of $mcfES\mathrm{slice}(\mathscr{E}_i^{\max}, \varphi)$ is a distributed lattice and is unique.

Further, let $\bigcup_{\mathscr{E}}^{\mathbf{C}} = \bigcup_{1 \le i \le N_{\mathscr{E}}} (mcf\mathrm{Conf}(mcfESslice)((\mathscr{E}_i^{\max}, \varphi)))$; $\bigcup_{\mathscr{E}}^{\mathbf{C}}$ is also unique and $(\bigcup_{\mathscr{E}}^{\mathbf{C}}, \subseteq)$ is a finitary coherent prime algebraic domain.

Next, we show that the slicing operation will keep the second part of necessary and sufficient condition for composition.

For any $\mathscr{E}_i^{\max}$ and $\mathscr{E}_j^{\max}$ $(j \ne i)$, if $E_{\mathscr{E}_i^{\max}} \cap E_{\mathscr{E}_j^{\max}} = D$ and $D \ne \emptyset$, we then have that $cf\mathrm{subConf}F(\mathscr{E}_i^{\max}, D) = cf\mathrm{subConf}F(\mathscr{E}_j^{\max}, D)$; that is, for any nonempty event subset D' $(D' \subseteq D)$ and any predicate φ, if any configuration of D' satisfies φ, that is, D' is the common part of both slices of $\mathscr{E}_i^{\max}$ and $\mathscr{E}_j^{\max}$. We still get that $cf\mathrm{subConf}F(\mathscr{E}_i^{\max}, D') = cf\mathrm{subConf}F(\mathscr{E}_j^{\max}, D')$.

This means that, for any two *mcfESs*, if their intersection is nonempty, no matter which part of the intersection belongs to the slice, after slicing, the necessary and sufficient condition for composition will be still satisfied.

Thus, we get that $\bigcup_{\mathscr{E}}^{\mathbf{C}}$ is a valid family of configurations for prime event structures; there should exist such a unique prime event structure $R \in \mathbb{E}$ that satisfies $\mathrm{Conf}F(R) = \bigcup_{\mathscr{E}}^{\mathbf{C}}$. We can get R by applying *wcc* to the corresponding event structures of $\bigcup_{\mathscr{E}}^{\mathbf{C}}$.

Therefore, the existence and uniqueness for event structure slicing have been established. We will then prove that the prime event structure R is the ultimate result of slicing.

(2) Satisfactoriness and Minimality. On the one hand, for any configuration X of event structure $\mathscr{E}$ that makes predicate φ hold, that is, $X \in \mathrm{Conf}F(\mathrm{Slice}ES(\mathscr{E}, \varphi))$, there must be a *mcfES*: $\mathscr{E}_i^{\max}$ so that $X \in \mathrm{Conf}F(\mathrm{Slice}ES(\mathscr{E}, \varphi))$;

let $\mathbf{C}_i = mcf\text{Conf}F(mcfES\text{slice}(\mathcal{E}_j^{\max}, \varphi))$, because $\mathbf{C}_i$ contains all the configurations of event structure $\mathcal{E}_j^{\max}$ that make predicate φ hold. We have that X must be contained by $\mathbf{C}_i$; that is, $X \in \mathbf{C}_i$.

We get $X \in \text{Conf}F(\text{Slice}ES(\mathcal{E}, \varphi)) \Rightarrow X \in \bigcup_{\mathcal{E}}^{\mathbf{C}}$.

Further, we get $\text{Conf}F(\text{Slice}ES(\mathcal{E}, \varphi)) \subseteq \bigcup_{\mathcal{E}}^{\mathbf{C}}$.

On the other hand, for any configuration $X \in \bigcup_{\mathcal{E}}^{\mathbf{C}}$, we get $X \in \text{Conf}F(\mathcal{E})$ and X can make predicate φ hold; then $X \in \text{Conf}F(\text{Slice}ES(\mathcal{E}))$ must hold. Thus, we get $X \in \bigcup_{\mathcal{E}}^{\mathbf{C}} \Rightarrow X \in \text{Conf}F(\text{Slice}ES(\mathcal{E}, \varphi))$.

That is, $\bigcup_{\mathcal{E}}^{\mathbf{C}} \subseteq \text{Conf}F(\text{Slice}ES(\mathcal{E}, \varphi))$.

Therefore, we have $\bigcup_{\mathcal{E}}^{\mathbf{C}} = \text{Conf}F(\text{Slice}ES(\mathcal{E}, \varphi))$.

Thus, we get that $\text{Slice}ES(\mathcal{E}, \varphi) = R$.

Moreover, by the definition of slice of maximal conflict-free event substructure, we have that, for any $\mathcal{E}_i^{\max}$ ($i \in \mathbf{N}, 1 \leq i \leq N_{\mathcal{E}}$), the corresponding $mcfES\text{slice}(\mathcal{E}_i^{\max}, \varphi)$ contains the least number of configurations that satisfy the given predicate φ; we then have that $\bigcup_{\mathcal{E}}^{\mathbf{C}} = \text{Conf}F(\text{Slice}ES(\mathcal{E}, \varphi))$ also contains the least number of configurations satisfying this specification. Thus, satisfactoriness and minimality both hold.

Consequently, from both (1) and (2), we conclude that the theorem holds.

5.3. Slicing Reduction Algorithm. In this section, we will present an approach for event structure slice computing. The slicing algorithm for a prime event structure or its *mcfESs* with respect to regular predicates is based on the *Adding Edges Theorem* (see [8, 20, 31, 33]).

In fact, by the following theorem, these lattices will never be actually constructed in the slicing process for efficiency.

The configurations do not satisfy the predicate but still can be included to complete the sublattice.

Given a distributive lattice L generated by a graph G, every sublattice of L can be generated by a graph obtained by adding edges to G. The following theorem holds.

Theorem 37 (Adding Edges Theorem). *Let L' be any sublattice of a finite distributive lattice L generated by the directed graph G. Then, there exists a graph G' that can be obtained by adding edges to (removing vertices from) G that generates L'.*

For any prime event structure, we can get the slices of its *mcfESs* by applying the Adding Edges Theorem. These slices can be composed by *wcc* to form a new prime event structure which is the target slice of the original event structure. This approach is less general but results in more efficient detection algorithms for a special class of predicates. Note that we will never actually construct the lattice or family of configurations of the event structure due to efficiency.

Garg and Mittal have presented an efficient algorithm $\text{slice}(G, \varphi)$ [8, 28] based on graphical representation G to compute the slice of POTs (or conflict-free event structures) with respect to a predicate φ. The algorithm adopts the principle of the Adding Edges Theorem and can produce a sliced graph representation. Especially, we have $G = \text{slice}(G, \text{true})$ for predicate $\varphi = \text{true}$ itself.

We extend the idea and algorithm to more general models and provide an algorithm for slicing the *mcfESs* and the original prime event structure. Thus, we have Algorithm 2 to compute the slice of conflict-free event structure.

For a prime event structure with conflict relations, we have to apply *cfp* operator to get $mcfsetN(\mathcal{E})$ maximal conflict-free event substructures and each of them can be sliced by *mcfES*slice. Then, the set consisting of each sliced result can be composed together by *wcc* to construct a new event structure. This new event structure will be the sliced result.

Thus, we can derive Algorithm 3 to compute the slice of a prime event structure.

Because the set of the slices of *mcfESs* may no longer keep the weak conflict relation which exists in the original *mcfESs*.

Therefore, after $mcfES\text{slice}(mcfES(\mathcal{E}), \varphi)$ operation is performed, the relation among these slices can be one of the following cases:

(1) strong conflict;

(2) conflict-free;

(3) weak conflict;

(4) hybrid of weak conflict and conflict-free;

(5) hybrid of weak conflict and strong conflict;

(6) hybrid of strong conflict, weak conflict, and conflict-free.

In case of (1), (3), and (5), the operation *wcc* can be performed directly. But in case of (2), (4), and (6), we have to add some events in order to make the result set of slices still be able to form a valid weak conflict set at the end of process.

For temporal predicates [8], such as EF, EG and AG can be computed by $\text{slice}(G, EF(\varphi))$, $\text{slice}(G, EG(\varphi))$, and $\text{slice}(G, AG(\varphi))$, respectively. From the definition of a slice, we know that every configuration of a slice $\text{slice}ES(\mathcal{E}, \varphi)$ is also a configuration of $\mathcal{E}$.

Clearly, the following two corollaries hold.

Corollary 38. *(1) For any prime event structure $\mathcal{E} \in \mathbb{E}$, $\text{Conf}F(\text{slice}ES(\mathcal{E}, EG(\varphi))) \subseteq \text{Conf}F(\text{slice}ES(\mathcal{E}, \varphi)) \subseteq \text{Conf}F(\mathcal{E})$.*

Similarly for AG, the following holds.

(2) For any prime event structure $\mathcal{E} \in \mathbb{E}$, $\text{Conf}F(\text{slice}ES(\mathcal{E}, AG(\varphi))) \subseteq \text{Conf}F(\text{slice}ES(\mathcal{E}, \varphi)) \subseteq \text{Conf}F(\mathcal{E})$.

Corollary 39. *For any prime event structure $\mathcal{E} \in \mathbb{E}$, $\text{Conf}F(\text{slice}ES(\mathcal{E}, EF(\varphi))) \subseteq \text{Conf}F(\mathcal{E})$.*

5.4. Case Study for Slicing Reduction. In this section, we will give an example to illustrate the prime event structure slice notion and its computing process.

Consider a prime event structure: $\mathcal{E} = (E_{\mathcal{E}}, \leq_{\mathcal{E}}, \sharp_{\mathcal{E}}, l_{\mathcal{E}})$, as shown in Figure 2. The components are described as follows:

(1) event set: $E_{\mathcal{E}} = \{e_a, e_b, e_c, e_1, e_2, e_3, e_4\}$;

(2) conflict relation: $\#_{\mathcal{E}} = \{e_b \# e_1\}$;

```
**Input**:
(1) a conflict-free event structure: $mcfES(\mathscr{E})$,
(2) a regular predicate: $\varphi$
**Output**: the slice of $mcfES$: ${\mathscr{E}_i^{\max}}_{slice}$
**BEGIN**
(1) $K = \emptyset$;
(2)     generate graph representation $G$ for $\mathscr{E}_i^{\max}$;
(3)     computing slice: $K = slice(G, \varphi)$;
(4)     generate event structure ${\mathscr{E}_i^{\max}}_{slice}$ from graph representation $K$;
(5)     return ${\mathscr{E}_i^{\max}}_{slice}$;
**END**
```

Algorithm 2: Slicing algorithm: $mcfESslice(mcfES(\mathscr{E}), \varphi)$.

```
**Input**:
(1) an event structure: $\mathscr{E}$
(2) a regular predicate: $\varphi$
**Output**: the slice: $\mathscr{E}_{slice}$
**BEGIN**
(1)  $int\ n = 0, C = \emptyset$;
(2)  $\mathscr{R} = \emptyset$;
(3)  $\mathscr{B}^{mcfES}(\mathscr{E}) = cfp(\mathscr{E})$
(4)  $n = mcfsetN(\mathscr{E})$;
(5)  for ($int\ i = 0; i < n; i{+}{+}$) {
(6)        get the $i$th $mcfES$: $\mathscr{E}_i^{\max}$ from $\mathscr{B}^{mcfES}$;
(7)        $C = C \cup mcfESslice(\mathscr{E}_i^{\max}, \varphi)$;
(8)  }
(9)  if ($C \neq \emptyset$){
(10)       $\mathscr{R} = wcc(C)$;
(11) }
(12) return $\mathscr{R}$;
**END**
```

Algorithm 3: Slicing algorithm: $sliceES(\mathscr{E}, \varphi)$.

(3) casual relation: $\preceq_{\mathscr{E}} = \{e_b \preceq e_a, e_c \preceq e_a, e_1 \preceq e_2, e_1 \preceq e_4, e_3 \preceq e_4, e_c \preceq e_3\}$;

(4) action labels: $l_{\mathscr{E}}(e_a) = a$, $l_{\mathscr{E}}(e_b) = b$, $l_{\mathscr{E}}(e_c) = c$, $l_{\mathscr{E}}(e_1) = a_1$, $l_{\mathscr{E}}(e_2) = a_2$, $l_{\mathscr{E}}(e_3) = a_3$, $l_{\mathscr{E}}(e_4) = a_4$;

(5) action functions:

(i) action(a_1) : $\{x = x + 1\}$;
(ii) action(a_2) : $\{x = x + 3\}$;
(iii) action(a_3) : $\{y = y + 3\}$;
(iv) action(a_4) : $\{y = y + 2\}$;
(v) action(a) : $\{z = z + 2\}$;
(vi) action(b) : $\{z = z + 1\}$;
(vii) action(c) : $\{y = y - 1\}$;
(viii) initValue : $\{x = 1; y = 1; z = 0\}$;

(6) slice criterion: $\{\varphi = -2 \leq (y - x + z) < 2\}$.

In this example, the system global states will be updated after an action function executes. Figure 2 depicts all the events conflict (for simplicity, only immediate conflict relation is shown) and casual relation. Figure 3 shows its corresponding family of configurations.

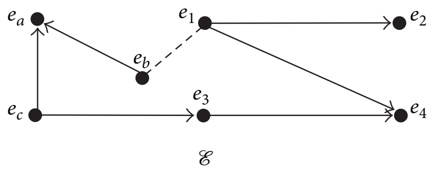
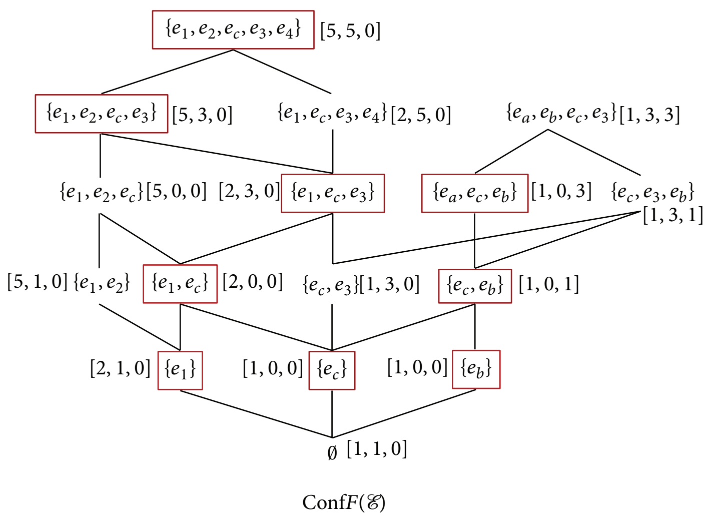


Figure 2: A Prime event structure $\mathscr{E}$.

There is one conflict relation between event b and e_1; due to the conflict inheritance property of prime event structures, we have $e_b \sharp e_1, e_b \sharp e_2, e_b \sharp e_4$ and $e_a \sharp e_1, e_a \sharp e_2, e_a \sharp e_4$; that is, each of $\{e_a, e_b\}$ is in conflict with each of $\{e_1, e_2, e_4\}$. Obviously, according to this conflict relation, apply cfp operation to the prime event structure $\mathscr{E}$ and we get that this event structure has only two $mcfESs$: they are $\mathscr{E}_1 = (\{e_a, e_b, e_c, e_3\}, \leq_{\mathscr{E}_1}, \emptyset, l_{\mathscr{E}_1})$ and $\mathscr{E}_2 = (\{e_1, e_2, e_3, e_4, e_c\}, \leq_{\mathscr{E}_2}, \emptyset, l_{\mathscr{E}_2})$ depicted by Figures



$\{e_1, e_2, e_c, e_3, e_4\}$ [5, 5, 0]
$\{e_1, e_2, e_c, e_3\}$ [5, 3, 0]
$\{e_1, e_c, e_3, e_4\}$ [2, 5, 0]
$\{e_a, e_b, e_c, e_3\}$ [1, 3, 3]
$\{e_1, e_2, e_c\}$ [5, 0, 0]
[2, 3, 0] $\{e_1, e_c, e_3\}$
$\{e_a, e_c, e_b\}$ [1, 0, 3]
$\{e_c, e_3, e_b\}$ [1, 3, 1]
[5, 1, 0] $\{e_1, e_2\}$
$\{e_1, e_c\}$ [2, 0, 0]
$\{e_c, e_3\}$ [1, 3, 0]
$\{e_c, e_b\}$ [1, 0, 1]
[2, 1, 0] $\{e_1\}$
[1, 0, 0] $\{e_c\}$
[1, 0, 0] $\{e_b\}$
$\emptyset$ [1, 1, 0]
Conf$F(\mathscr{E})$

FIGURE 3: Family of configurations of $\mathscr{E}$.

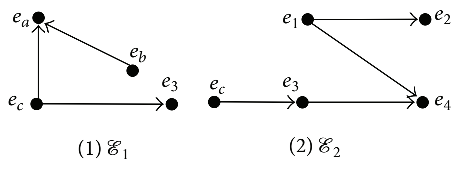


FIGURE 4: *mcfESs* of $\mathscr{E}$: $\mathscr{E}_1$ and $\mathscr{E}_2$.

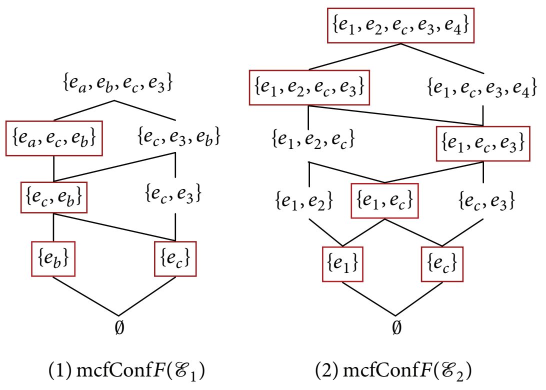


FIGURE 5: Families of configurations of *mcfESs*($\mathscr{E}_1$ and $\mathscr{E}_2$).

4(1) and 4(2), respectively, and Figures 5(1) and 5(2) show their corresponding families of configurations.

The configurations that satisfy the predicate are labelled with frames. In fact, these configurations are only used to describe relationship between original event structure and its slice graphically; in general, they will never be actually constructed in the slicing algorithm for efficiency.

The families of configurations of the slices of $\mathscr{E}_1$ and $\mathscr{E}_2$ with respect to the predicate $\{\varphi = -2 \le (y - x + z) < 2\}$ are shown in Figures 6(1) and 6(2), respectively. It can be verified that both *mcf*Conf($\mathscr{E}_1$) and *mcf*Conf($\mathscr{E}_2$) form distributed lattices.

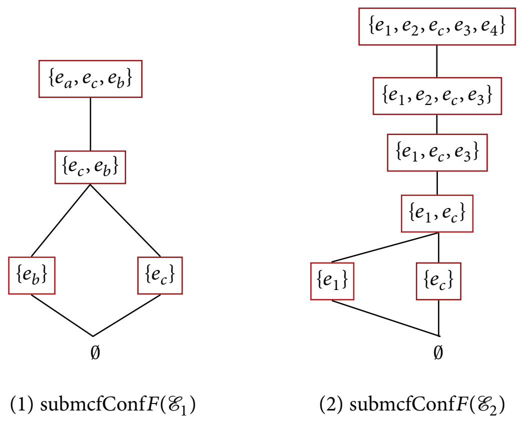


FIGURE 6: Subfamilies of configurations of the slices.

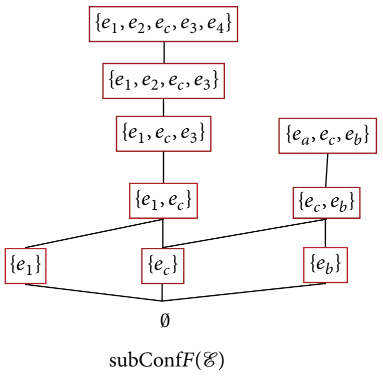


FIGURE 7: Subfamily of configurations of the slice.

These configurations of the slices are exactly the ones that satisfy the given predicate in the family of configurations of the original event structure. Figure 7 shows the family of configurations constructed by applying $\cup$ operation to the families of configurations of all slices.

Finally, in Figure 8, the slice of $\mathscr{E}_1$ and the slice of $\mathscr{E}_2$ are combined into the slice of $\mathscr{E}$ by *wcc* operator, as expected.

To illustrate the benefit of predicate detection by using slicing reduction as shown in above example, consider the states in Figure 3 again.

Let $\{\varphi = -2 \leq (y - x + z) < 2\}$ be the predicate to be checked, and suppose we want to detect whether $EF(\varphi)$ holds or not; that is, there exists a global state that satisfies φ. Without slicing reduction applied, we are forced to examine all global states, 15 states in total as shown in Figure 3, to decide whether the traces satisfy the predicate.

Alternatively, we can compute the slice via slicing reduction technique with respect to the regular temporal predicate and use this slice for predicate detection.

For this purpose, firstly, we compute the slice with respect to φ and the slice is shown in Figure 7.

Finally, we check whether the initial state is the same as the initial state of the slice and decide whether the predicate is satisfied or not.

The slice contains only 9 states and has much fewer states than the original traces itself. Generally, it is exponentially smaller in many cases and this can result in substantial savings.

6. Symmetry Reduction

Finite state systems frequently exhibit symmetry which can be found in memories, caches, register files, bus protocols, and anything that has a lot of replicated structures. The use of symmetry to reduce state space has been investigated widely by researchers [2, 3, 15, 22, 23].

In this section, we will discuss symmetry properties over prime event structures. Symmetry in an event structure implies the existence of nontrivial permutation groups that preserve both the events labelling and all relations of causal dependence and independence that exist between events. We start by introducing some notions of group theory.

6.1. Automorphism Groups. We know that the set of all permutation on a set forms a permutation group under functional composition. A permutation group over a finite set X consists of bijections, $X \rightarrow X$, and their compositions as the binary operations.

Definition 40 (permutation group). Let X be a finite set; a *permutation* of X is a bijection from X to itself. Then, $S_X := \{f : X \rightarrow X \mid f \textit{ is abijection}\}$; that is, the family of all the permutations of the set X, denoted by S_X, forms a group called the *symmetric group* on X. For any bijection $f \in S_X$ is called a *permutation.* Any subgroup of is called a permutation group on X.

Obviously, a symmetric group is a special permutation group. A permutation group over a set has good properties; specially, it can induce an equivalence relation. The equivalence classes of an equivalence relation on a set can form a partition of this set. Thus, for a set X, if there exists a permutation group on the set X, the permutation group can induce a partition of the set X. We can easily check this property.

In this paper, permutation groups are used to partition the set of events in an event structure so that we use equivalence classes (orbits) of events to investigate symmetry in this event structure.

Definition 41 (automorphism). Let $\mathscr{E} = (E, \preceq, \sharp, l) \in \mathbb{E}$ and let G be a permutation group on the event set E of $\mathscr{E}$. A permutation $f \in G$ is said to be an *automorphism* of E if and only if f satisfies the following conditions:

(1) $\forall e_1, e_2 \in E : e_1 \preceq e_2 \Rightarrow f(e_1) \preceq f(e_2)$, $e_1 \sharp e_2 \Rightarrow f(e_1) \sharp f(e_2)$;

(2) $l(e_1) = l(f(e_2))$.

Definition 42 (automorphism group). A permutation group G is called an *automorphism group* for the event structure $\mathscr{E}$ ($\mathscr{E} = (E, \preceq, \sharp, l) \in \mathbb{E}$) if and only if every permutation $f \in G$ is an automorphism of $\mathscr{E}$.

Notice that every $f \in G$ has an inverse, which is also an automorphism; our definition of an automorphism group can prove that $f \in G$ is an automorphism for an event structure $\mathscr{E}$ if and only if f satisfies the following condition: $\forall e_1, e_2 \in E$: $e_1 \preceq e_2 \Leftrightarrow f(e_1) \preceq f(e_2); e_1 \sharp e_2 \Leftrightarrow f(e_1) \sharp f(e_2)$; and $l(e_1) = l(f(e_2))$.

6.2. Quotient Model of an Event Structure. The symmetric quotient model for an event structure is a structural reduced model.

Let G be a permutation group acting on the set E and $e \in E$; then the orbit of e is the set $\theta(e) = \{d \mid \exists g \in G : f(e) = d\}$. From each orbit $\theta(e)$ we pick a representative that is called $\mathrm{rep}(\theta(e))$. Intuitively, the quotient model can be obtained by collapsing all the events to orbits.

Definition 43 (symmetric quotient model). Let $\mathscr{E} = (E, \preceq, \sharp, l) \in \mathbb{E}$ and let G be an automorphism group on the event set E of the event structure $\mathscr{E}$. The *symmetric quotient model* $\mathscr{E}_G = (E_G, \preceq_G, \sharp_G, l_G)$ is defined as follows.

(1) The event set is $E_G = \{\theta(e) \mid e \in E\}$, the set of orbits of the events in E.

(2) The causality relation $\preceq_G$ is given by $\preceq_G = \{(\theta(e_1), \theta(e_2)) \mid (e_1, e_2) \in \preceq\}$ and the inverse of $\preceq_G$ is denoted by $\succeq_G$.

(3) The conflict relation $\sharp_G$ is given by $\sharp_G = E_G \times E_G \setminus (\prec_G \cup \succ_G \cup \ \mathrm{co}_G \cup \{(e'', e'') \mid e'' \in E_G\})$, where $\mathrm{co}_G = \{(\theta(e_1), \theta(e_2)) \mid \mathrm{co} = E \times E(\preceq \cup \succeq \cup \sharp \cup (e', e') \mid e' \in E), (e_1, e_2) \in E\}$.

(4) The labelling function l_G is given by $l_G(\theta(e)) = l(\mathrm{rep}(\theta(e)))$.

An automorphism group G of an event structure $\mathscr{E} = (E, \preceq, \sharp, l)$ is an invariance group for an action $a \in Act$ if and only if the following condition holds: $\forall f \in G, e \in E : l(\theta) = a \Leftrightarrow l(f(\theta)) = a$.

We then say that a is an invariant under G. Thus, if G is an invariance group for all actions in the action set Act of $\mathscr{E}$ and $\mathscr{E}_G$ is the symmetric quotient model for $\mathscr{E}$; we can directly have $\forall a \in \text{Act}, e \in E : l(\theta) \Leftrightarrow l_G(\theta(e)) = l(\text{rep}(\theta(e)) = a$.

From the above definition, we have that the symmetric quotient model is still a prime event structure and preserves all the causal dependence and independence relations of the original, but the conflict relations are reduced. Note that every two different events in an orbit are exactly in conflict with each other. The quotient model can preserve all the behavior of the original one.

6.3. Symmetry Reduction Algorithm. Based on previous discussion, we have Algorithm 4 for symmetry reduction. In this algorithm, replicated substructure will be removed and only one copy will be kept.

Firstly, cfp operator will be applied on a given event structure to get a set of conflict-free substructures.

Then, for any two elements of them, a checking procedure will be performed to remove the redundant one.

Finally, wcc will be applied on the substructure set to get the reduced model. Detailed pseudocode description is shown in Algorithm 4.

6.4. An Example for Symmetry Reduction. In this section, an example will be discussed to demonstrate the reduction process based on Algorithm 4.

The example of a prime event structure $\mathscr{E}$ with five events (e_1, e_2, e_3, e_4, and e_5) and its semantics in terms of families of configurations is given in Figures 9(1) and 9(2), respectively.

The action-labelling (action set: $Act = \{a, b, c, d\}$) function is defined as follows: $l(e_1) = a$, $l(e_2) = b$, $l(e_3) = c$, $l(e_4) = b$, and $l(e_5) = d$.

In this structure, we have $e_3 \preceq e_5$, $e_2 \sharp e_4$, $e_3 \sharp e_2$, and $e_3 \sharp e_4$. We also have $e_5 \sharp e_2$, $e_5 \sharp e_1$, and $e_5 \sharp e_4$(due to conflict inheritance).

Then event set is $E_G = \{e_1, e_2, e_3, e_4, e_5\}$ and we can construct a permutation group G on the set E_G where two permutations are as follows:

$$\begin{pmatrix} e_1 & e_2 & e_3 & e_4 & e_5 \\ e_1 & e_2 & e_3 & e_4 & e_5 \end{pmatrix} \begin{pmatrix} e_1 & e_2 & e_3 & e_4 & e_5 \\ e_1 & e_4 & e_3 & e_2 & e_5 \end{pmatrix}. \tag{3}$$

Obviously, the group G is an invariance group for all actions in $Act = \{a, b, c, d\}$. We then have the orbits of events in being $\theta(e_2) = \theta(e_4) = \{e_2, e_4\}$, $\theta(e_1) = \{e_1\}$, $\theta(e_3) = \{e_3\}$, and $\theta(e_5) = \{e_5\}$. Thus, the symmetric quotient model can be described as shown in Figure 10.

We can get weak conflict set of the event structure $\mathscr{E}$ by cfp operator as follows: $cfp(\mathscr{E}) = \{\mathscr{E}_1, \mathscr{E}_2, \mathscr{E}_3\}$, where

$$\begin{aligned} \mathscr{E}_1 &= (E_1, \preceq_1, \sharp_1, l_1), \\ \mathscr{E}_2 &= (E_2, \preceq_2, \sharp_2, l_2), \\ \mathscr{E}_3 &= (E_3, \preceq_3, \sharp_3, l_3), \\ E_1 &= \{e_1, e_2\}, \\ E_2 &= \{e_1, e_4\}, \\ E_3 &= \{e_1, e_3, e_5\}. \end{aligned} \tag{4}$$

Thus, we have $E_1 \sharp^w E_2$, $E_1 \sharp^w E_3$, and $E_3 \sharp^w E_2$ (or, $\mathscr{E}_1 \sharp^w \mathscr{E}_2$, $\mathscr{E}_1 \sharp^w \mathscr{E}_3$, and $\mathscr{E}_3 \sharp^w \mathscr{E}_2$).

By definition, we have that $\mathscr{E}_1$ and $\mathscr{E}_2$ are symmetric. According to the symmetry reduction algorithm, we will remove replicated events but keep the common or representation ones. The substructure $\mathscr{E}_2$ is removed. The resulted substructure set consists of two elements: $\mathscr{E}_1$ and $\mathscr{E}_3$, because $\mathscr{E}_1$ and $\mathscr{E}_3$ are weak conflict sets of $\mathscr{E}$ and wcc operator can be applied on them to construct a new event structure: $\mathscr{E}' = wcc(\mathscr{E}_1, \mathscr{E}_3)$. Thus, the symmetric reduced event structure can be described by Figure 11.

To illustrate the advantage for properties checking by using symmetry reduction as shown in the above example, consider the states in Figure 9, 9 states in total.

Suppose we want to check whether a property $EF(p)$ holds or not; that is, there exists a global state that satisfies φ. Without symmetry reduction applied, we have to examine all global states, 9 states in total. But with symmetry reduction, as shown in Figure 11 or Figure 10 to decide whether the property holds, only 7 states should be checked and a considerable saving can be achieved.

7. Mathematical Framework

In this section, we will provide a unified mathematical framework for slicing and symmetry reduction based on cfp and wcc operators.

Firstly, we will review some basic definitions in this section. Here, we refer the reader to [14] for details. Next, we will introduce the single action transitions [3] for event structures. Finally, we will discuss the related theories for slicing and symmetric reduction and establish the unified framework.

7.1. Basic Definitions

Definition 44. For any event structure $\mathscr{E} = (E_{\mathscr{E}}, \preceq_{\mathscr{E}}, \sharp_{\mathscr{E}}, l_{\mathscr{E}}) \in \mathbb{E}$, define $\mathscr{L}[\mathscr{E}] = (\text{Conf}F(\mathscr{E}), \subseteq)$. Especially, for any conflict-free event structure $\mathscr{F} = (E_{\mathscr{F}}, \preceq_{\mathscr{F}}, \emptyset, l_{\mathscr{F}}) \in \mathbb{F}$, define $\mathscr{L}[\mathscr{F}] = (\text{Conf}F(\mathscr{F}), \subseteq)$.

In fact, $\mathscr{L}[\mathscr{F}]$ is the partial order of left-closed and conflict-free subsets of $E_{\mathscr{E}}$ ordered by set inclusion.

$\mathscr{L}[\mathscr{E}]$ is the partial order of left-closed subsets of $E_{\mathscr{F}}$ ordered by set inclusion.

Definition 45 (single action transition). Let $\mathscr{E} \in \mathbb{E}$. A transition $X \overset{a}{\rightarrow}_{\mathscr{E}} X'$ is called a *single action transition* if and only if $a \in Act, X, X' \in \text{Conf}F(\mathscr{E}), X \subseteq X'$ and there exists an event e such that $X' - X = e$ with $l_{\mathscr{E}}(e) = a$.

Here, $X \overset{a}{\rightarrow}_{\mathscr{E}} X'$ indicates that in the event structure the state represented by the configuration X may evolve into a state represented by the configuration by performing

```
Input: a prime event structure $\mathscr{E}$;
Output: the reduced model of the event structure: $\mathscr{E}_{sr}$;
BEGIN
(1)  $C = \emptyset; i = 1; n = 0; B = \emptyset; \mathscr{B} = \emptyset$;
     /* Step One: partition via $cfp$ operator, we will get all maximal conflict-free sub-structure of an event structure */
(2)     $B = \mathscr{B}^{mcfES}(\mathscr{E}) = cfp(\mathscr{E})$;
(3)     $n = mcfsetN(\mathscr{E})$;
     /* Step Two: automorphism checking for sub-structure */
(4)   for ($i = 0; i \le n; i{+}{+}$) {
(5)      for ($j = 0; i \le i; j{+}{+}$) {
     /* Step Two-1: get action set of each maximal conflict-free sub-structure from $B$ */
(6)        $L_{Act}(E_i) = \{a \in Act \mid \exists e \in E_i : l(e) = a\}$
(7)        $L_{Act}(E_j) = \{a \in Act \mid \exists e \in E_j : l(e) = a\}$
     /* $IsNotIdentical$: checking two sets are are identical or not */
     /* $|S|$: the element amount of the set $S$ */
     /* Step Two-2: checking their action sets are identical or not */
(8)        if (($|L_{Act}(E_i| \ne |L_{Act}(E_j)|$) ||
             $IsNotIdentical(L_{Act}(E_i), L_{Act}(E_j))$ {
(9)        break; }
     /* Step Two-3: checking their causal relations are identical or not */
(10)        if (($|\preceq_{\mathscr{E}_i^{\max}}| \ne |\preceq_{\mathscr{E}_j^{\max}}|$) ||
             $IsNotIdentical(\preceq_{\mathscr{E}_i^{\max}}, \preceq_{\mathscr{E}_j^{\max}})$ {
(11)        break; }
     /* Step Two-4: checking their conflict relations are identical or not */
(12)        if (($|\#_{\mathscr{E}_i^{\max}}| \ne |\#_{\mathscr{E}_j^{\max}}|$) ||
             $IsNotIdentical(\#_{\mathscr{E}_i^{\max}}, \#_{\mathscr{E}_j^{\max}})$ {
(13)        break; }
     /* Step Three: automorphism exists, remove the duplicated one and merge for reduction */
(14)        $B = B - \mathscr{E}_j^{\max}$;
(15)        $n = n - 1$;
(16)        } /* end for $j$ */
(17)    } /* end for $i$ */
(18) return $\mathscr{E}_{sr} = wcc(B)$;
END;
```

Algorithm 4: Symmetry reduction: $sr(\mathscr{E})$.

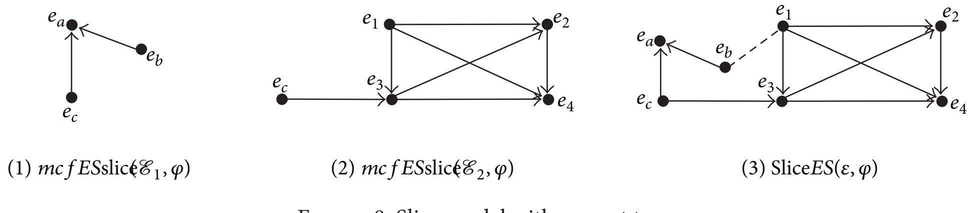

e_a
e_b
e_c
(1) $mcfES$slice$(\mathscr{E}_1, \varphi)$
e_1
e_2
e_3
e_4
e_c
(2) $mcfES$slice$(\mathscr{E}_2, \varphi)$
e_1
e_2
e_a
e_b
e_3
e_c
e_4
(3) Slice$ES(\varepsilon, \varphi)$

Figure 8: Slice model with respect to φ.

the action a. This transition relation associates a labelled system based on single action transitions with each event structure.

Definition 46 (trace). Let $\mathscr{E} \in \mathbb{E}$. A word $w = a_1 \dots a_n \in Act^*$ is called *trace* of $\mathscr{E}$ if and only if $\exists X_0, \dots, X_n \in \mathrm{Conf}F(\mathscr{E})$: $X_0 = \emptyset$ and $X_{i-1} \xrightarrow{a_i} X_i,\ i = 1, \dots, n$.

Here, let $\mathrm{Trs}(\mathscr{E})$ denote the set of all traces of $\mathscr{E}$.

Definition 47 (interleave trace equivalence). Let $\mathscr{E}_1, \mathscr{E}_2 \in \mathbb{E}$. $\mathscr{E}_1$ and $\mathscr{E}_2$ are called *interleave trace equivalence* ($\mathscr{E}_1 \approx_{it} \mathscr{E}_2$) if and only if $\mathrm{Trs}(\mathscr{E}_1) = \mathrm{Trs}(\mathscr{E}_2)$.

Definition 48 (interleaving bisimulation). Let $\mathscr{E}_1, \mathscr{E}_2 \in \mathbb{E}$. A relation $R \subseteq C(\mathscr{E}_1) \times C(\mathscr{E}_2)$ is called *interleaving bisimulation*

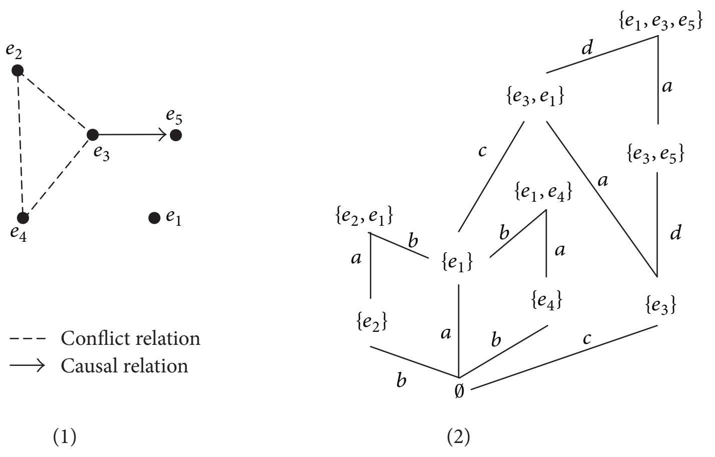


Figure 9: An event structure and its family of configurations.

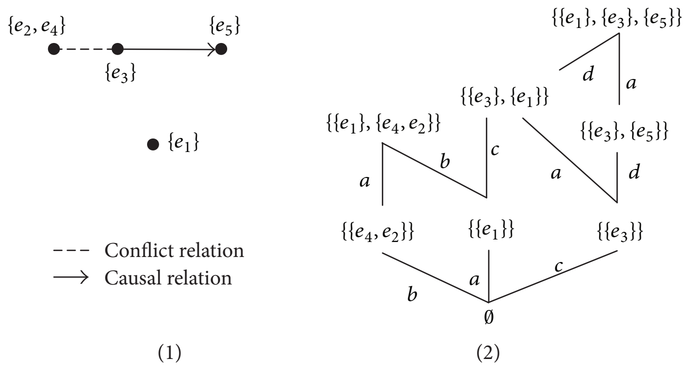


Figure 10: Symmetric quotient model of $\mathscr{E}$: $\mathscr{E}_G$.

between $\mathscr{E}_1$ and $\mathscr{E}_2$ if and only if $(\emptyset, \emptyset) \in R$ and $(X, Y) \in R$; then

(1) $X \xrightarrow{a}_{\mathscr{E}_1} X'$, $a \in Act \Rightarrow \exists Y' : Y \xrightarrow{a}_{\mathscr{E}_2} Y' \wedge (X', Y') \in R$;

(2) $Y \xrightarrow{a}_{\mathscr{E}_2} Y'$, $a \in Act \Rightarrow \exists X' : X \xrightarrow{a}_{\mathscr{E}_1} X' \wedge (X', Y') \in R$.

Definition 49 (interleaving bisimulation equivalent). Let $\mathscr{E}_1, \mathscr{E}_2 \in \mathbb{E}$. $\mathscr{E}_1$ and $\mathscr{E}_2$ are called *interleaving bisimulation equivalent* ($\mathscr{E}_1 \approx_{ib} \mathscr{E}_2$) if and only if there exists an interleaving bisimulation between $\mathscr{E}_1$ and $\mathscr{E}_2$.

7.2. Unified Framework. The unified framework for slicing and symmetry reduction we have set up can be pictured as in Figure 12.

(1) Isomorphism and Equivalence. We are concerned with the translation of concepts and ideas from one side to the other. The following theorems hold.

Theorem 50. *Let $\mathscr{F}$ be a conflict-free event structure, $\mathscr{F} = (E_{\mathscr{F}}, \preceq_{\mathscr{F}}, \emptyset, l_{\mathscr{F}}) \in \mathbb{F}$; then $\mathscr{F} \cong \wp[\mathscr{L}[\mathscr{F}]]$. Similarly, let $P = (D, \sqsubseteq)$ be a prime algebraic complete lattice; then $P \cong \mathscr{L}[\wp[P]]$.*

Theorem 51. *Let $\mathscr{E}$ be a prime event structure, $\mathscr{E} = (E_{\mathscr{E}}, \preceq_{\mathscr{E}}, \sharp_{\mathscr{E}}, l_{\mathscr{E}}) \in \mathbb{E}$; then $\mathscr{E} \cong \wp[\mathscr{L}[\mathscr{E}]]$. Similarly, let $P = (D, \sqsubseteq)$ be a prime algebraic coherent domain; then $P \cong \mathscr{L}[\wp[P]]$.*

Clearly, from Theorems 50 and 51, we have the following.

(1) For any prime event structure $\mathscr{E}$, we have that $\mathscr{E} \cong \wp[\mathscr{L}[\mathscr{E}]]$.

(2) Similarly, for any partial order $P = (D, \sqsubseteq)$, P is a prime algebraic complete lattice or a finitary coherent prime algebraic domain; we have that $P \cong \mathscr{L}[\wp[P]]$.

From Theorems 23, 24, and 25, we have that conflict-free event structures and prime algebraic complete lattices are equivalent; this implies that there is a one-to-one correspondence between a prime event structure and its family of configurations. Similarly, prime event structures and finitary coherent prime algebraic domains are also equivalent; one can be used to represent the other.

(2) Mutual Inverse Operation. For any prime event structure, *cfp* and *wcc* are mutually inverse operators.

We can get the full set of maximal conflict-free substructures of a prime event structure by *cfp* operator.

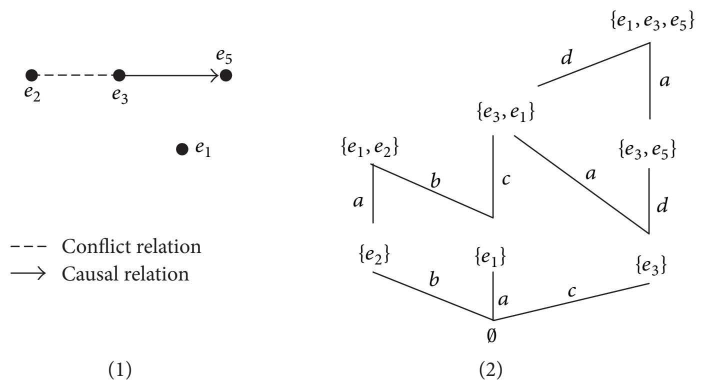
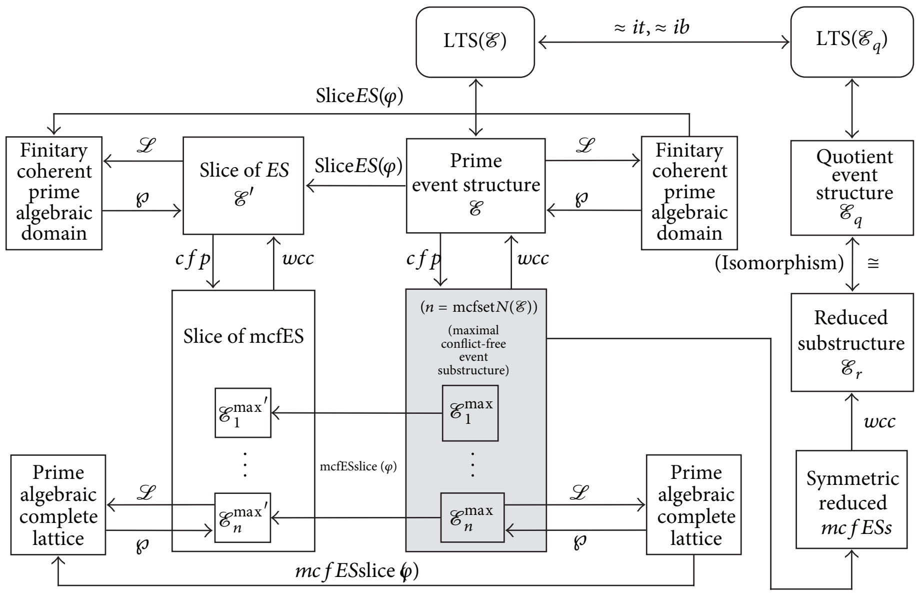


FIGURE 11: Reduced model: $\mathscr{E}'$.



LTS($\mathscr{E}$)
$\approx it, \approx ib$
LTS($\mathscr{E}_q$)
Slice$ES(\varphi)$
Finitary coherent prime algebraic domain
$\mathscr{L}$
$\wp$
Slice of ES $\mathscr{E}'$
Prime event structure $\mathscr{E}$
Quotient event structure $\mathscr{E}_q$
cfp
wcc
(Isomorphism) $\cong$
Slice of mcfES
(n = mcfset$N(\mathscr{E})$)
(maximal conflict-free event substructure)
$\mathscr{E}_1^{\max'}$
$\mathscr{E}_n^{\max'}$
$\mathscr{E}_1^{\max}$
$\mathscr{E}_n^{\max}$
mcfESslice (φ)
Reduced substructure $\mathscr{E}_r$
Symmetric reduced $mcfESs$
Prime algebraic complete lattice
$mcfES$slice (φ)

FIGURE 12: Unified mathematical framework.

Conversely, given the full set of maximal conflict-free substructures of the prime event structure, we can recover the original prime event structure by *wcc* operator.

(3) Slicing Reduction. Firstly, compared with traditional computation slicing, Figure 12 demonstrates the translation between a conflict-free event structure (or its slice) and prime algebraic complete lattice. This forms the theoretical basis of computation slicing technique proposed by Garg and Mittal, Sen [7, 20].

Secondly, Figure 12 also demonstrates the translation between a prime event structure (or its slice) and finitary coherent prime algebraic domain, which serves as the theoretical basis of our event structure slicing with conflict. A pair of mutually inverse operators, *cfp* and *wcc*, act as a link between the two parts.

Thirdly, the slice of a prime event structure can be computed by the following steps:

(1) applying *cfp* operator to partition the original prime event structure into a full set of maximal conflict-free substructures;

(2) for each maximal conflict-free substructure, applying traditional computation slicing algorithm based on Adding Edges Theorem [7, 8, 20] over its directed graph representation to compute the slice with respect to the given predicate;

(3) applying *wcc* operator to compose the resulted slices in above step (2) and form a new prime event structure, while the generated event structure is the slice of the original prime event structure.

(4) Symmetry Reduction. Operators *cfp* and *wcc* can also play an important role in symmetry reduction.

Symmetry reduction of a prime event structure can be performed by the following steps:

(1) applying *cfp* operator to partition the original prime event structure into a full set of maximal conflict-free substructures;

(2) checking automorphism among the produced substructures and checking causal relation and action set;

(3) removing duplicated substructure and applying *wcc* operator to compose the resulted structure to form a newly generated prime event structure, which will be symmetry reduced prime event structure.

(5) Trace Equivalence. The relation between original event structure and its quotient model can be specified by Theorems 52 and 53 (See [3]).

Theorem 52. *Let* $\mathscr{E} \in \mathbb{E}$ *and* $\mathscr{E}_G$ *be quotient structure of* $\mathscr{E}$*; then* $\mathscr{E}_G \approx_{it} \mathscr{E}$.

Theorem 53. *Let* $\mathscr{E} \in \mathbb{E}$ *and let* $\mathscr{E}_G$ *be the symmetric quotient model for* $\mathscr{E}$*. Then* $\mathscr{E} \simeq_{ib} \mathscr{E}_G$.

Generally, given an event structure $\mathscr{E}$, in fact, its behavior is exhibited by the labelled transition system (LTS, for short) $\mathrm{LTS}_{\mathscr{E}} = \{\mathrm{Conf}F(\mathscr{E}), Act_{\mathscr{E}}, T_{\mathscr{E}}, \emptyset\}$, where

(1) the configurations $\mathrm{Conf}F(\mathscr{E})$ are states;

(2) the set of labels is the set of actions $Act_{\mathscr{E}}$;

(3) the transitions $T_{\mathscr{E}}$ are single action transitions between every two configurations of $\mathscr{E}$; namely, $T_{\mathscr{E}} \subseteq \mathrm{Conf}F(\mathscr{E}) \times Act_{\mathscr{E}} \times \mathrm{Conf}F(\mathscr{E})$;

(4) the initial configuration (the empty set $\emptyset$) is the initial state.

The above equivalence is based on labelled transition systems whose transitions are single action transitions. As shown in Figure 12, we construct labelled transition system for prime event structure and its quotient model, we will have that their LTSs are interleaving bisimulation and interleave trace equivalence also. Therefore, the following corollary holds.

Corollary 54. *Let* $\mathscr{E} \in \mathbb{E}$ *and* $\mathscr{E}_G$ *let be the symmetric quotient model for* $\mathscr{E}$*. If* $LTS = (ConfF(\mathscr{E}), Act, \rightarrow_{\mathscr{E}}, \emptyset)$ *and* $LTS_G = (ConfF(\mathscr{E}_G), Act, \rightarrow_{\mathscr{E}_G}, \emptyset)$ *are induced from* $\mathscr{E}$ *and* $\mathscr{E}_G$*, respectively, then* $LTS \approx_{it} LTS_G$ *and* $LTS \approx_{ib} LTS_G$ *hold.*

Evidently, we have the following theorems.

Theorem 55. *For any* $\mathscr{E} \in \mathbb{E}$*, its symmetric quotient model* $\mathscr{E}_G$ *and symmetric reduced model* $\mathscr{E}'$ *are isomorphism; that is,* $\mathscr{E}_G \cong \mathscr{E}'$.

Proof. It is not difficult to prove it by constructing a bijection between the events and their orbits.

Theorem 56. *For any* $\mathscr{E} \in \mathbb{E}$*, its symmetric reduced model* $\mathscr{E}'$ *is a substructure of* $\mathscr{E}$*; that is,* $\mathscr{E}' \lhd \mathscr{E}$.

Proof. The proof is straightforward.

(6) Technique Combination. Symmetry reduction technique is orthogonal to slicing reduction and can be used in conjunction with slicing. Thus, it is easy to have the following result.

Theorem 57. *For any* $\mathscr{E} \in \mathbb{E}$*, let* φ *be a regular predicate; then the following statements hold:* $sr(SliceES(\mathscr{E}, \varphi)) = wcc(mcfESslice(cfp(sr(\mathscr{E})), \varphi))$ *and* $sr(SliceES(\mathscr{E}, \varphi)) = SliceES(sr(\mathscr{E}), \varphi)$.

Proof. The proof is straightforward.

8. Conclusion

In this paper, we presented a unified mathematical framework for event structure slicing and symmetric reduction. We described the equivalent relationship between original event structure and its maximal conflict-free event substructures. We proposed two mutually inverse operators: conflict-free partition operator and weak choice composition operator. Both symmetry reduction and slicing reduction can be performed by this pair of operators. We also investigated the related properties, translations, and correspondences between event structures and domains. Essentially, slicing over event structure is a high level extension to the traditional computation slicing based on the model with conflict.

Slicing technique can make the verification of program behavior easier by reducing the size of the state space to be analyzed. Symmetry reduce is another powerful structural reduction technique that can also be applied to narrow down state space. Both quotient model and sliced model produced by reduction are often much smaller than the original model. The consequential model can be used to significantly improve the effectiveness of property verification of the original model.

In future work, on the one hand, we will extend our work to other more complicated event structure models, such as flow event structure [34] and bundle event structure [35, 36]. On the other hand, we hope to implement and test our approach on various verification tools in practice. In addition, we would like to exploit more possible applications and theories.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

The project is supported by the Fundamental Research Funds for the Central Universities (DUT14QY05) (1600-851025). The authors would like to thank their colleagues for participating in the research. They also appreciate the anonymous reviewers for their helpful comments.

References

[1] S. Alagar and S. Venkatesan, "Techniques to tackle state explosion in global predicate detection," *IEEE Transactions on Software Engineering*, vol. 27, no. 8, pp. 704–714, 2001.

[2] E. M. Clarke, R. Enders, T. Filkorn, and S. Jha, "Exploiting symmetry in temporal logic model checking," *Formal Methods in System Design*, vol. 9, pp. 77–104, 1996.

[3] J. Jiang and J. Wu, "Symmetry and autobisimulation," in *Proceedings of the 6th International Conference on Parallel and Distributed Computing, Applications and Technologies (PDCAT '05)*, pp. 866–870, IEEE Computer Society Press, December 2005.

[4] J.-Z. Wu and H. Fecher, "Symmetric structure in logic programming," *Journal of Computer Science and Technology*, vol. 19, no. 6, pp. 803–811, 2004.

[5] M. Weiser, *Program slices: formal, psychological, and practical investigations of an automatic program abstraction method [Ph.D. thesis]*, University of Michigan, 1979.

[6] M. Weiser, "Programmers use slices when debugging," *Communications of the ACM*, vol. 25, no. 7, pp. 446–452, 1982.

[7] V. K. Garg and N. Mittal, "On slicing a distributed computation," in *Proceedings of the 21st IEEE International Conference on Distributed Computing Systems (ICDCS '01)*, D. Harel, D. Kozen, and J. Tiuryn, Eds., Dynamic Logic, pp. 322–329, MIT Press, Phoenix, Ariz, USA, 2001.

[8] A. Sen and V. K. Garg, "Formal verification of simulation traces using computation slicing," *IEEE Transactions on Computers*, vol. 56, no. 4, pp. 511–527, 2007.

[9] E. Duesterwald, R. Gupta, and M. L. Soffa, "Distributed slicing and partial re-execution for distributed programs," in *Proceedings of the 5th Workshop on Language and Compilers for Parallel Computing*, pp. 329–337, 1992.

[10] J. W. de Bakker, W. P. de Roever, and G. Rozenberg, Eds., *Linear Time, Branching Time and Partial Order in Logics and Models for Concurrency*, vol. 354 of *Lecture Notes in Computer Science*, Springer, Berlin, Germany, 1989.

[11] G. Winskel, "Event structures," in *Petri Nets: Applications and Relationships to Other Models of Concurrency, Advances in Petri Nets 1986, Part II; Proceedings of an Advanced Course, Bad Honnef, September 1986*, W. Brauer, W. Reisig, and G. Rozenberg, Eds., vol. 255 of *Lecture Notes in Computer Science*, pp. 325–392, Springer, Berlin, Germany, 1987.

[12] P. Madhusudan, "Model-checking trace event structures," in *Proceedings of the 18th Annual IEEE Symposium on Logic in Computer Science (LICS '03)*, pp. 371–380, June 2003.

[13] M. Mukund and P. S. Thiagarajan, "An axiomatization of event structures," in *Foundations of Software Technology and Theoretical Computer Science (Bangalore, 1989)*, vol. 405 of *Lecture Notes in Computer Science*, pp. 143–160, Springer, Berlin, Germany, 1989.

[14] R. J. van Glabbeek and U. Goltz, "Refinement of actions and equivalence notions for concurrent systems," Hildesheimer Informatik-Bericht 6/98, University of Hildesheim, 1998.

[15] H. Hermanns and M. Ribaudo, "Exploiting symmetries in stochastic process algebras," in *Proceedings of the European Simulation Multiconference (ESM '98)*, pp. 763–770, SCS Europe, 1998.

[16] M. Nielsen, G. Plotkin, and G. Winskel, "Petri nets, event structures and domains. I," *Theoretical Computer Science*, vol. 13, no. 1, pp. 85–108, 1981.

[17] M. Nielsen, G. Plotkin, and G. Winskel, "Petri nets, event structures and domains. I," *Theoretical Computer Science*, vol. 13, no. 1, pp. 85–108, 1981.

[18] R. Loogen and U. Goltz, "Modelling nondeterministic concurrent processes with event structures," *Fundamenta Informaticae*, vol. 14, no. 1, pp. 39–73, 1991.

[19] A. Sen, J. Bhadra, V. K. Garg, and J. A. Abraham, "Formal verification of a system-on-chip using computation slicing," in *Proceedings of the International Test Conference (ITC '04)*, pp. 810–819, October 2004.

[20] A. Sen, *Techniques for formal verification of concurrent and distributed program traces [Ph.D. thesis]*, The University of Texas at Austin, Austin, Tex, USA, 2004, http://www.library.utexas.edu/etd/d/2004/senma042/senma042.pdf.

[21] E. A. Emerson and A. P. Sistla, "Symmetry and model checking," in *Computer Aided Verification (Elounda, 1993)*, vol. 697 of *Lecture Notes in Computer Science*, pp. 463–478, Springer, Berlin, Germany, 1993.

[22] A. Miller, A. Donaldson, and M. Calder, "Symmetry in temporal logic model checking," *ACM Computing Surveys*, vol. 38, no. 3, article 8, 2006.

[23] G. Winskel, "Event structures with symmetry," in *Computation, Meaning, and Logic: Articles Dedicated to Gordon Plotkin*, vol. 172 of *Electronic Notes in Theoretical Computer Science*, pp. 611–652, Elsevier, Amsterdam, The Netherlands, 2007.

[24] X. Gao, J. Wu, R. Qiao, and J. Chen, "Theory framework for event structure slicing," in *Proceedings of the 13th IEEE Symposium on Computers and Communications (ISCC '08)*, pp. 714–721, IEEE, Marrakech, Morocco, July 2008.

[25] G. Winskel, "An introduction to event structures," in *Linear Time, Branching Time and Partial Order in Logics and Models for Concurrency (Noordwijkerhout, 1988)*, J. W. de Bakker, W. P. de Roever, and G. Rozenberg, Eds., vol. 354 of *Lecture Notes in Computer Science*, pp. 364–397, Springer, Berlin, Germany, 1989.

[26] G. Winskel, "An introduction to event structures," in *Linear Time, Branching Time and Partial Order in Logics and Models for Concurrency (Noordwijkerhout, 1988)*, vol. 354 of *Lecture Notes in Computer Science*, pp. 364–397, Springer, Berlin, Germany, 1989.

[27] G. Winskel and M. Nielsen, "Models for concurrency," in *Handbook of Logic in Computer Science*, vol. 4, pp. 1–148, Oxford University Press, New York, NY, USA, 1995.

[28] N. Mittal and V. K. Garg, "On detecting global predicates in distributed computations," in *Proceedings of the 21st IEEE International Conference on Distributed Computing Systems*, pp. 3–10, Phoenix, Ariz, USA, April 2001.

[29] V. K. Garg, "Algorithmic combinatorics based on slicing posets," *Theoretical Computer Science*, vol. 359, no. 1–3, pp. 200–213, 2006.

[30] E. M. Clarke, E. A. Emerson, and A. P. Sistla, "Automatic verification of finite state concurrent systems using temporal logic," *ACM Transactions on Programming Languages and Systems*, vol. 8, no. 2, pp. 244–263, 1986.

[31] B. A. Davey and H. A. Priestley, *Introduction to Lattices and Order*, Cambridge Mathematical Textbooks, Cambridge University Press, Cambridge, UK, 1990.

[32] A. Sen and V. K. Garg, "Detecting temporal logic predicates in distributed programs using computation slicing," in *Proceedings of the 7th International Conference on Principles of Distributed Systems (OPODIS '03)*, December 2003.

[33] N. Mittal, A. Sen, V. K. Garg, and R. Atreya, "Finding satisfying global states: all for one and one for all," in *Proceedings of the 18th International Parallel and Distributed Processing Symposium (IPDPS '04)*, Santa Fe, NM, USA, April 2004.

[34] G. Boudol, "Flow event structures and flow nets," in *Semantics of Systems of Concurrent Processes (La Roche Posay, 1990)*, vol. 469 of *Lecture Notes in Computer Science*, pp. 62–95, Springer, Berlin, Germany, 1990.

[35] H. Fecher, M. Majster-Cederbaum, and J. Wu, "Bundle event structures: a revised cpo approach," *Information Processing Letters*, vol. 83, no. 1, pp. 7–12, 2002.

[36] R. Langerak, "Bundle event structures: a non-interleaving semantics for LOTOS," in *Formal Description Techniques V*, M. Diaz and R. Groz, Eds., pp. 331–346, 1992.

Hindawi Publishing Corporation
Journal of Applied Mathematics
Volume 2014, Article ID 703098, 12 pages
http://dx.doi.org/10.1155/2014/703098

Research Article

Compositional Abstraction Refinement for Component-Based Systems

Lianyi Zhang, Qingdi Meng, and Kueiming Lo

School of Software, Tsinghua University, Beijing 100084, China

Correspondence should be addressed to Lianyi Zhang; lyzhang117@gmail.com

Received 14 March 2014; Accepted 9 May 2014; Published 11 June 2014

Academic Editor: Xiaoyu Song

The efficiency of the compositional verification of invariants depends on the abstraction, which may lead to verification incompleteness. The invariant strengthening and state partitioning techniques are proposed in this paper. The former could refine the overapproximation by removing the unreachable states, and the latter is a variant of counterexample-guided abstraction refinement. Integrated with these two refinement techniques, a unified compositional verification framework is presented to strengthen the abstraction and find counterexamples. Some examples are included to show that the verification of the safety properties in component-based systems has been achieved by our framework.

1. Introduction

A component-based system is made up of several components using a set of glue (interactions) that describes the coordination between components [1]. Some components, like program processes, contain local data and control information to determine their own behaviors, and their interactions represent the communication between components. Component-based system has the advantages that the development can be simplified and the time-to-market would be reduced. However, the tremendously high (or even infinite) number of states of the components makes the verification of properties of these systems intractable. In this paper, we discuss the compositional verification using abstraction refinement techniques on component-based systems.

Compositional verification would alleviate the problem of state explosion in concurrent model checking, especially in conjunction with abstraction. The work in [2, 3] proposed an automated compositional abstraction framework which differs in the communication method. The abstraction procedure in [2] is guided by the data (interaction through shared variables) of the model while it is guided by the action (interaction through message-passing event) in [3]. Thread modular reasoning is also a compositional abstraction framework proposed in [4], which automatically generates the environment for each thread by analyzing its interaction. Compositional verification using abstraction refinement techniques on component-based systems has been extensively studied. Malkis et al. combined CEGAR and thread modular reasoning in their proposed technique [5], called thread-modular CEGAR. The technique computed reachable states with Cartesian abstraction. They directly compute the reachable states for all processes of the compositional system in an explicit way. Then refinement was applied to eliminate unreachable states by removing them from the Cartesian product. A technique closely related to compositional verification is assume-guarantee reasoning. Henzinger et al. [6, 7] proposed a component-based verification algorithm which is complete for verifying safety properties. They first initialize each component as true (the most abstract model which essentially accepts any behaviors) and then iteratively refine them by adding new auxiliary predicates, whereas they initialize the guarantee of the components as false (the most abstract model which essentially rejects any behaviors) and refine them successively by considering abstraction of current component and guarantees of other components.

Another interesting branch for compositional verification is the inductive invariant based method [8–10]. The properties established from the verification are usually invariants.

Given a property on component-based system states, compositional verification of invariant is to verify whether the given property can be inferred from the invariants of their components. Divide-and-conquer approaches can be applied in compositional verification to infer global invariants from local invariants. A compositional verification method based on the inductive invariant is presented in [11, 12]. In this approach, the property of the component-based system can be inferred by the compositional verification rule which is referred to as the component invariant and the interaction invariant. This approach is incomplete. Given a property (predicate), if the conjunction of the component invariants and the interaction invariants is able to establish the property, one concludes the verification. Otherwise, one reports that the verification is inconclusive.

The inconclusive problem is due to two reasons. One is the *abstraction*, which transforms the original system (infinite) to an abstract system (finite). The other is the *interaction invariant*, which is the overapproximation of the reachable abstract states. Thus, the conjunction of component invariants and the interaction invariant is such an overapproximation that unreachable states in the original system may be computed as reachable in the abstract system, which may include unreachable error states. We propose two refinement techniques, counterexample-guided invariant strengthening and state partitioning to address the inconclusive problem.

A unified compositional abstraction refinement algorithm for the invariant checking problem on component-based systems is given, which contains two phases. At the verification phase, we apply compositional verification to the invariant checking problem. If it suffices to conclude the verification, we are done. Otherwise, the verification algorithm moves to the refinement phases. In the other phase, we first strengthen the interaction invariant and remove the spurious counterexamples. After the invariant is strengthened, we analyze the remaining counterexamples and partition abstract states. When our algorithm returns to the verification phase, added states will induce a refined abstract model.

The rest is organized as follows. The component-based system, component invariant, and component abstraction are introduced in Section 2. In Section 3, we introduce the compositional verification of invariant for component-based systems, which is the overapproximation of the system states. We give counterexample-guided invariant strength and partition refinement methods in Section 4. A compositional verification framework integrated with iterative refinement and its correctness proof are presented in Section 5. We show the effectiveness of our proposed method in Section 6 and then conclude in Section 7.

2. Preliminaries

In this section, we present concepts and definitions that are used in this paper. We start with the overview of component-based systems, which includes the concepts of atomic component, interaction, and compound component. In the rest of this section, we introduce the concepts for the component invariant and abstraction.

2.1. Component-Based System. Component-based system is a basic model of wide-ranging concurrent systems. It is proven in [13] that such a hierarchical model can be converted to semantically equivalent flat model. Therefore, instead of analyzing complex hierarchical structure, we concentrate on two-level structure in this paper. The lower level of the component-based system is atomic component, and the higher level is compound component which is composed of several atomic components with a set of interactions.

An *atomic component* is a transition system, denoted by a tuple $B = (L, P, T, X, \{g_t\}_{t\in T}, \{f_t\}_{t\in T})$, where (L, P, T) is a transition system; that is, $L = \{l_1, l_2, \dots, l_k\}$ is a set of control locations, P is a set of ports, and $T \subseteq L \times P \times L$ is a set of transitions. $X = \{x_1, \dots, x_n\}$ is a set of variables and for each $t \in T$, respectively, $g_t(\mathbf{x})$ is a guard, a predicate on $\mathbf{x}$, and $f_t(\mathbf{x}, \mathbf{x}')$ is an update relation, a formula on $\mathbf{x}$(current) and $\mathbf{x}'$(next) state variables. Given a transition $t = (l, p, l') \in T$, l and l' are, respectively, the source and target location denoted by $\bullet t$ and $t\bullet$.

Given a set of components $B_1, B_2, \dots, B_n$ where $B_i = (L_i, P_i, T_i, X_i, \{g_t\}_{t\in T_i}, \{f_t\}_{t\in T_i})$, an interaction a is a set of ports, subset of $\bigcup_{i=1}^{n} P_i$, such that, for all $i = 1, \dots, n$ s.t. $|a \cap P_i| \le 1$. The trans(a) is a set of transitions $\{t_i = (l_i, p_i, l_i') \in T_i\}_{i\in I}$, for arbitrary $I \subseteq \{1, \dots, n\}$, which have to execute synchronously.

A compound component $\gamma(B_1, \dots, B_n)$ is also a transition system, denoted by $B = (L, \gamma, T, X, \{g_t\}_{t\in T}, \{f_t\}_{t\in T})$, where

(1) (L, γ, T) is the transition system such that

  (i) $L = L_1 \times L_2 \times \cdots \times L_n$ is a set of control locations,

  (ii) $T \subseteq L \times \gamma \times L$ contains transitions $t = ((l_1, \dots, l_n), a, (l_1', \dots, l_n'))$ obtained by synchronization of sets of transitions $\{t_i = (l_i, p_i, l_i') \in T_i\}_{i\in I}$ such that $\{p_i\}_{i\in I} = a \in \gamma$ and $l_j' = l_j$ if $j \notin I$, for arbitrary $I \subseteq \{1, \dots, n\}$;

(2) $X = \bigcup_{i=1}^{n} X_i$ and for $a \in \gamma$, the transition trans(a) resulting from the synchronization of a set of transitions $\{t_i\}_{i\in I}$, the associated guard and update predicate are, respectively, $g_t = \bigwedge_{i\in I} g_{t_i}$ and $f_t = \bigwedge_{i\in I} f_{t_i} \wedge \bigwedge_{i\notin I}(X_i' = X_i)$.

A typical transition system (e.g., a component B) contains control states, usually the program counter which takes values from the set of control locations. Control states are defined by formulas of the form (at_l_i), where $l_i \in L$. Additionally, we assume that for each transition $t = (l_i, p, l_j) \in T$, we get a transition formula written as at_$l_i \wedge g_t(\mathbf{x}) \mapsto f_t(\mathbf{x}, \mathbf{x}')$; at_$l_j$ where $\mathbf{x} \in X$.

Definition 1 (component system). A component system is denoted by $\mathcal{S} = (B, \text{Init})$, where B is a component (atomic or compound component) and Init is the initial predicate of the system.

2.2. Component Invariant. Most properties established during verification are either invariants or depend on invariants. To prove that a predicate ϕ is an invariant of some system $\mathcal{S}$, we need to find such a predicate ψ that ψ is stronger than ϕ and ψ is inductive. In general, for a system S, ψ is an invariant of S if every initial state of system S satisfies ψ and ψ is preserved under all transitions of the system.

Definition 2 (component invariant). Given a component $B = (L, P, T, X, \{g_t\}_{t\in T}, \{f_t\}_{t\in T})$, for a global state predicate $\Phi = \bigvee_{l\in L} \text{at\_}l \wedge \varphi_l$, the generated formula by post predicate is $\text{post}(\Phi) = \bigvee_{l\in L}(\bigvee_{t=(l,p,l')} \text{at\_ } l' \wedge \text{post}_t(\varphi_l))$, where $\text{post}_t(\varphi(X)) = \exists X' \cdot (g_t(X') \wedge f_t(X', X) \wedge \varphi(X'))$. A fixed point of the post transformer is a component invariant (formula Φ) of the component B; that is, $\text{post}(\Phi) \Leftrightarrow \Phi$.

The component invariant is computed by the post predicate transformer. A more detailed technique, as well as other related properties for component invariants, is given in [8, 14] and will not be presented.

Consider a system $\mathcal{S} = (B, \text{Init})$, where B is a component and Init is a state predicate satisfied by the initial states of B. Φ is an invariant of S, if it is a component invariant of B and $\text{Init} \Rightarrow \Phi$.

2.3. Component Abstraction. When original system has real or large range discrete variables, enumeration of concrete states by valuation is impossible. Before computing interaction invariants of the compound system, we need first to generate a finite state abstraction for the component-based system.

In our methodology, given a component-based system $\mathcal{S} = \langle B, \text{Init}\rangle$, component invariant $\Phi = \bigvee_{l\in L} \text{at\_}l \wedge (\bigvee_{m\in M_l} \varphi_{lm})$ is automatically generated from a transition system (a component) that describes its behavior, which is represented by the disjunction of atomic formulas, such as $\text{at\_}l \wedge \varphi_{lm}$. However, instead of using the atomic formulas to generate an abstract system, we use them to construct an *abstraction function*.

The abstraction function maps an atomic formula to an abstract value of that formula, which preserves the relation among the atomic formulas instead of treating them as independent propositions. The abstraction technique used here is based on the approach proposed by Bensalem et al. in [2, 8]. An abstraction function α maps the concrete representation of the atomic predicates $\text{at\_}l \wedge \varphi_{lm}$ to the abstract presentation ϕ, called abstract state. We use Φ^α to represent a set of abstract states.

Definition 3 (abstract component). Given a component B, a component invariant Φ of B, and the associated abstraction function α, the abstract component B^α is denoted by $B^\alpha = (\Phi^\alpha, P, \rightsquigarrow)$, where

(i) Φ^α is the abstract state, where Φ is the component invariant and α is abstraction function,

(ii) P is a set of ports of B,

(iii) $\rightsquigarrow$ is the transition of B. For any pair of abstract states $\phi = \alpha(\text{at\_}l \wedge \varphi)$ and $\phi' = \alpha(\text{at\_}l' \wedge \varphi')$, there exists the transition from ϕ to ϕ' through p (denoted by $\phi \overset{p}{\rightsquigarrow} \phi'$) if and only if $\exists t = (l, p, l') \in T$, $\text{post}_t(\varphi) \wedge \varphi' \neq \text{false}$.

3. Compositional Verification

After the overview of component-based system, the component invariant, and the component abstraction, we first present the compositional verification rule of component-based systems, then we give out the concepts of the verification rules, and last we do the analysis on the overapproximation of the compositional abstraction.

3.1. Compositional Verification Rule. In the approach presented in [11, 12], the property Φ of $\gamma(B_1, \ldots, B_n)$ can be inferred by the following compositional verification rule:

$$\frac{\{B_i \langle \Phi_i \rangle\}_i^n, \Psi \in \text{II}\left(\{B_1^{\alpha_1}, \ldots, B_n^{\alpha_n}\}, \gamma\right), \left(\bigwedge_i^n \Phi_i\right) \wedge \Psi \Rightarrow \Phi}{\gamma\left(B_1, \ldots, B_n\right)\langle \Phi \rangle}, \tag{1}$$

where $B_i\langle\Phi_i\rangle$ means that Φ_i is an invariant of component B_i and Ψ belongs to the set of interaction invariants II. For component B_i, a predicate Φ_i on the component states is a component invariant of B_i. Before generating the interaction invariants, one needs to compute the abstraction for the components. The abstraction for B_i, denoted by $B_i^{\alpha_i}$, is computed from Φ_i by the abstraction function α_i. In the abstract system $\gamma(\{B_1^{\alpha_1}, \ldots, B_n^{\alpha_n}\})$, we generate the interaction invariant Ψ under the global behavior constraint, which is the overapproximation of the reachable abstract states. Given a property (predicate) Φ, if the conjunction of the invariant $\bigwedge_i^n \Phi_i$ and the interaction invariants Ψ are able to establish the property, one concludes the verification. Otherwise, one reports that the verification is inconclusive.

3.2. Related Concepts. Consider a compound system $\mathcal{S} = \langle\gamma(B_1, \ldots, B_n), \text{Init}\rangle$ and a set of component invariants $\Phi_1, \ldots, \Phi_n$ associated with the atomic components. If $B_i^{\alpha_i}$ is an abstraction of B_i with respect to an invariant Φ_i and its abstraction function α_i for $i = 1, \ldots, n$, then $B^\alpha = \gamma(B_1^{\alpha_1}, \ldots, B_n^{\alpha_n})$ is an abstraction of $B = \gamma(B_1, \ldots, B_n)$ with respect to $\bigwedge_{i=1}^n \Phi_i$ and an abstraction function α obtained as the composition of the α_i.

Definition 4 (abstract system). Given a component system $\mathcal{S} = (B, \text{Init})$, the abstract system is denoted by $\mathcal{S}^\alpha = \langle B^\alpha, \text{Init}^\alpha\rangle$, where B^α is the abstraction for a component and $\text{Init}^\alpha = \bigvee_{\phi\in\Phi_0^\alpha} \text{at\_}\phi$, where $\Phi_0^\alpha = \{\phi \in \Phi^\alpha \mid \alpha^{-1}(\phi) \wedge \text{Init} \neq \text{false}\}$ is the set of the initial abstract states.

The following lemma in [11] says that $\gamma(B_1^{\alpha_1}, \ldots, B_n^{\alpha_n})$ is an abstraction of $B = \gamma(B_1, \ldots, B_n)$ and that invariants of the abstract system are also invariants of the concrete system.

Lemma 5. *If B^α is an abstraction of B with respect to an invariant Φ and α its abstraction function, then B^α simulates*

B. Moreover, if Φ^α is an invariant of $\mathcal{S}^\alpha$, then $\alpha^{-1}(\Phi^\alpha)$ is an invariant of $\mathcal{S}$.

Given an abstract system, interaction invariants characterize constraints on the global abstract state space induced by synchronization among components. Interaction invariants are based on the concept of traps in Petri net, which are computed on finite transition system $\gamma(B_1^{\alpha_1}, \dots, B_n^{\alpha_n})$ without variables but abstract locations ϕ.

Consider a compound system $\gamma(B_1^{\alpha_1}, \dots, B_n^{\alpha_n})$, where $B_i^{\alpha_i} = (\Phi_i^{\alpha_i}, P_i, \rightsquigarrow_i)$ is a transition system. For a set of abstract states $\Phi^\alpha \subseteq \bigcup_{i=1}^n \Phi_i^{\alpha_i}$, the forward interaction set is denoted by $\Phi^\alpha{\scriptstyle\blacksquare} = \bigcup_{\phi \in \Phi^\alpha} \phi{\scriptstyle\blacksquare}$ where $\phi{\scriptstyle\blacksquare} = \{\{\tau_i\}_{i \in I} \mid \forall i \cdot \tau_i \in \rightsquigarrow_i \wedge \exists i \cdot {\scriptstyle\blacksquare}\tau_i = \phi \wedge \{\text{ port } (\tau_i)\}_{i \in I} \in \gamma\}$ is related to some interactions. In each interaction, there exists a transition τ_i from ϕ participate in. That is, $\phi{\scriptstyle\blacksquare}$ is composed of sets of transitions involved in some interaction of γ in which a transition τ_i from ϕ can participate.

Given a parallel composition $\gamma(B_1^{\alpha_1}, \dots, B_n^{\alpha_n})$, where $B_i^{\alpha_i} = (\Phi_i^{\alpha_i}, P_i, \rightsquigarrow_i)$, a trap [15] is a set Φ^α of abstract locations $\Phi^\alpha \subseteq \bigcup_{i=1}^n \Phi_i^{\alpha_i}$ such that $\Phi^\alpha{\scriptstyle\blacksquare} \subseteq {\scriptstyle\blacksquare}\Phi^\alpha$.

Definition 6 (interaction invariant). Given an abstract system $\mathcal{S}^\alpha = \langle \gamma(B_1^{\alpha_1}, \dots, B_n^{\alpha_n}), \text{Init}^\alpha \rangle$, if the set of locations $\Phi^\alpha \subseteq \bigcup_{i=1}^n \Phi_i^{\alpha_i}$ is a trap containing the initial states of some components, then $\bigvee_{\phi \in \Phi^\alpha} \text{at\_}\phi$ is the interaction invariant of $\mathcal{S}^\alpha$.

The characterization of traps in [12] allows one to compute the set of traps [16] using different approaches, including methods of positive mapping or fix-pointed computation.

3.3. Overapproximation. In the compositional verification rule present in Section 1, the conjunction of component invariants and the interaction invariant is an *overapproximation* of reachable system states. The overapproximation of the compositional verification mainly results from two kinds of abstractions. One is due to the *abstraction function*, and the other is caused by interaction interference. The conjunction of *interaction invariants* overapproximates the set of reachable states, which may cause some unreachable error states to be included.

The abstract system B^α for a component B is generated by elimination in a conservative way. To check whether $\phi \rightsquigarrow \phi'$, where $\phi = \alpha(\text{at\_}l \wedge \varphi)$ and $\phi' = \alpha(\text{at\_}l' \wedge \varphi')$, we check that for all transitions $t = (l, p, l')$ we have $\text{post}_t(\varphi) \wedge \varphi' = \text{false}$. When we generate abstract systems, the behaviors of the abstract system are more than the original system's. As the compositional verification performance on abstract components, the verification is incompleteness.

The conjunction of interaction invariant (a trap) characterizes constraints on the states induced by synchronization among components. States satisfying interaction invariants provide enabled execution information, a local projection of global states on part of components. However, these global states may not be reachable from initial states. Moreover, different interactions may include common ports. The interactions including the same port must exclusively execute, that is, just only one interaction can execute. Interaction invariant cannot characterize this dynamic information.

From the above analysis, we can conclude that invariant-based compositional verification is incomplete. We propose two refinement techniques in conjunction with invariant-based compositional verification rule to get a verification framework. The framework is an iterative scheme which starts from the abstraction until a concrete counterexample is found or until the safety property holds on abstract systems.

4. Refinement Approaches

We give counterexample-guided invariant strengthening and partition refinement methods in this section.

4.1. Invariant Strengthening. In the first part of this section, we present the invariant strengthening approach. As interaction invariant computed by the greatest fixed point is the overapproximation, counterexamples (represented by the conjunction of abstract states) may be spurious. For this case, we check whether counterexamples can be reachable from the initial states. If not, we generate a fixed point backward from the spurious counterexample and strengthen the invariant by the fixed point.

Given an abstract component system, $\mathcal{S}^\alpha = \langle B^\alpha, \text{Init}^\alpha \rangle$, where $B^\alpha = \gamma(B_1^{\alpha_1}, \dots, B_n^{\alpha_n})$ and $B_i^{\alpha_i} = (\Phi_i^{\alpha_i}, P_i, \rightsquigarrow_i)$. Formally, $\mathcal{S}^\alpha = (\Phi, \mathcal{T}, \text{Init}^\alpha)$ is a finite state machine (FSM) in which

(i) Φ is a set of global states; for the abstract state $\phi = (\phi_1, \dots, \phi_n)_{\phi_i \in \Phi_i^{\alpha_i}}$, we can represent it as conjunction (a vector) of local abstract state; that is, $\phi = \bigwedge_{\phi_i \in \Phi_i^{\alpha_i}} \phi_i \in \Phi$;

(ii) $\mathcal{T}$ is a predicate on global states Φ and Φ'; for $((\phi_1, \dots, \phi_n), a, (\phi_1', \dots, \phi_n'))$, where $a \in \gamma$, there exists $\mathcal{T}(\phi, \phi')$, where $\phi = \bigwedge_{\phi_i \in \Phi_i^{\alpha_i}} \phi_i$ and $\phi' = \bigwedge_{\phi_i' \in \Phi_i^{\alpha_i}} \phi_i'$;

(iii) Init^α is the initial predicate of the abstract system.

A formula φ is interpreted as the set $|[\varphi]|$ of all the global states; $\phi \in \Phi$ satisfy φ. We define the set $\text{Reach}_{\mathcal{T}}(\text{Init}^\alpha)$ of the global states *reachable* from the states $|[\text{Init}^\alpha]|$ via the transition as the smallest set such that $|[\text{Init}^\alpha]| \subseteq \text{Reach}_{\mathcal{T}}(\text{Init}^\alpha)$; and $\{\phi' \mid \exists \phi \cdot \mathcal{T}(\phi, \phi')\} \subseteq \text{Reach}_{\mathcal{T}}(\text{Init}^\alpha)$ if $\phi \subseteq \text{Reach}_{\mathcal{T}}(\text{Init}^\alpha)$. Consider an abstract component system $\mathcal{S}^\alpha = (\Phi, \mathcal{T}, \text{Init}^\alpha)$, where Φ is a set of the global states, $\mathcal{T}$ is the transition predicate, and Init^α is the initial predicate. The converse transition system $(\Phi, \mathcal{T}^{-1}, \text{Init}^\alpha)$ is defined by $\mathcal{T}^{-1}(\phi', \phi) = \mathcal{T}(\phi, \phi')$. For a formula φ, we compute all the successor states which are generated from the states hold φ by $\text{sp}_{\mathcal{T}}(\varphi)$. $\text{sp}_{\mathcal{T}}(\varphi) = \{\phi' \mid \exists \phi \cdot (\mathcal{T}(\phi, \phi') \wedge \varphi)\}$, where $\phi, \phi' \in \Phi$ and ϕ holds φ. Likewise, we can define $\text{sp}_{\mathcal{T}^{-1}}(\varphi) = \{\phi' \mid \exists \phi \cdot (\mathcal{T}(\phi', \phi) \wedge \varphi)\}$, where $\phi, \phi' \in \Phi$ and ϕ holds φ.

The following lemma [17] says that if none of the states represented by ϕ is backward reachable from the initial states, then $\neg\phi$ is an invariant.

Lemma 7. *Let $\mathcal{S}^\alpha = \langle \Phi, \mathcal{T}, Init^\alpha \rangle$ be a transition system and ϕ an arbitrary formula (or states). If φ is such that $(sp_{\mathcal{T}^{-1}}(\varphi) \vee$*

$\phi) \Rightarrow \varphi$ and the formula $Init \wedge \varphi$ is unsatisfiable, then $\neg\varphi$ is an inductive invariant of $\mathcal{S}^{\alpha}$.

Corollary 8. *If $Reach_{\mathcal{T}^{-1}}(\varphi) \cap Init = \emptyset$, then the formula corresponding to the complement of the set of $Reach_{\mathcal{T}^{-1}}(\varphi)$ is an inductive invariant.*

Theorem 9. *Assume that vilo is a counterexample and vilo $\wedge\ \alpha^{-1}(\phi) \neq$ false, where ϕ is an abstract state of $\mathcal{S}^{\alpha}$. If $Reach_{\mathcal{T}^{-1}}(\phi) \cap Init^{\alpha} = \emptyset$, then vilo is the spurious counterexample and the complement of the set of $Reach_{\mathcal{T}^{-1}}(\phi)$ is used to strengthen the invariants of S^{α}.*

Inspired by the above theorem, we propose an approach, called the invariant strengthening, to strengthen the invariants from the compositional verification rule.

We first choose a counterexample and generate a fixed point from the selected counterexample using backward propagation. An abstract state including a counterexample is the conjunction formula of abstract states like $\phi = \bigwedge_{\phi_i \in \Phi_i^{\alpha_i}} \phi_i$, where $\phi_i = \alpha_i(\text{at\_}l \wedge \varphi)$, which is the product of abstract states of each component. On the abstract system $\mathcal{S}^{\alpha}$, we do the backward propagation by $\text{sp}_{\mathcal{T}^{-1}}(\phi) = \{\phi' \mid \exists\phi \cdot (\mathcal{T}(\phi', \phi) \wedge \phi)\}$. Then from a counterexample, we can compute all the global abstract system states using backward propagation. If the intersection of the generated fixed point and the initial states is empty, the counterexample and all of the backward generated fixed points are not reachable from initial states. For this case, we say the selected counterexample is spurious. The invariant strengthening approach removes the spurious counterexample and eliminates the unreachable states generated from the spurious counterexample to strengthen the invariant.

For invariant strengthening, time consumes at counterexample reachable global states set computation. As this set computation is based on fixed-pointed computation and is monotonic, so time complexity for Algorithm 1 is $\mathcal{O}(\Phi)$, where Φ is number of global states of abstract system and $\Phi = \prod_{i=1}^{n} \Phi_i$ and $\Phi_i = |\Phi_i^{\alpha}|$ the number of abstract states from ith component.

4.2. Partition Refinement. In this part, we proposed a more effective counterexample-guided method, called partition refinement, which can accelerate abstraction refinement mechanism. Consider the abstraction function which transforms the original infinite system to the finite abstract system. An abstract state is an aggregation of a number of original system states. If a concrete state Φ_c is part of an abstract state Φ, we cannot make decisions (1) whether Φ is a counterexample, if Φ_c is a counterexample in concrete system; (2) whether Φ_c is reachable in concrete system, although Φ is reachable in abstract system. We propose our refinement approach to give the solution of the above question (Figure 1).

In the above approach, we analyze whether the remaining counterexamples are the spurious counterexamples which are caused by inaccurate characterization. To eliminate these spurious counterexamples, we refine the abstract component states by the proposed state partition approach. Consider an abstract system state $\phi^{\alpha} = \text{at\_}l \wedge \varphi$ and a counterexample $\phi_e = \text{at\_}l \wedge \varphi_e$ such that $\varphi_e \Rightarrow \varphi$ (Dec_0). We can partition ϕ^{α} into two states ϕ_e and ϕ', $\phi_e = \text{at\_}l \wedge \varphi_e$ and $\phi' = \text{at\_}l \wedge \varphi'$, where $\varphi' = \varphi - \varphi_e$; here we denote $\phi_e \vee \phi' = \phi^{\alpha}$ and $\phi_e \wedge \phi' =$ false. Assume that there exist ϕ_1^{α} and ϕ_2^{α}, from which ϕ^{α} is reachable such that $\text{post}_{\tau_1}(\alpha^{-1}(\phi_1^{\alpha}) \cdot \varphi) \wedge \alpha^{-1}(\phi_e) \cdot \varphi \neq$ false and $\text{post}_{\tau_2}(\alpha^{-1}(\phi_2^{\alpha}) \cdot \varphi) \wedge \alpha^{-1}(\phi') \cdot \varphi \neq$ false (Dec_1) hold. We remove ϕ^{α} from abstract state space Φ^{α} and add ϕ_e and ϕ' into Φ^{α}.

From the above, we assume that ϕ_1^{α} reach the error-part ϕ_e of ϕ^{α} and ϕ_2^{α} reach the other ϕ'. In this case, we should add the transitions into the system. If $\exists\phi_i^{\alpha}, \exists\tau_i : (\phi_i^{\alpha}, p_i, \phi^{\alpha}) \in \rightsquigarrow$ such that $\text{post}_{\tau_i}(\alpha^{-1}(\phi_i^{\alpha}) \cdot \varphi) \wedge \alpha^{-1}(\phi) \cdot \varphi \neq$ false (Dec_2), we add a transition from ϕ_i^{α} to ϕ (here, ϕ stands for both ϕ_e and ϕ'); likewise, if $\exists\phi_i^{\alpha}$, and $\exists\tau_i : (\phi^{\alpha}, p_i, \phi_i^{\alpha}) \in \rightsquigarrow$ such that $\text{post}_{\tau}(\alpha^{-1}(\phi) \cdot \varphi) \wedge \alpha^{-1}(\phi_i^{\alpha}) \cdot \varphi \neq$ false (Dec_3), we add a transition from ϕ to ϕ_i^{α}.

Theorem 10. *If $\mathcal{S}_{ref}^{\alpha}$ is a refined system returned by Algorithm 2, then B_{ref}^{α} simulates B. Moreover, if Φ^{α} is an invariant of B_{ref}^{α}, then $\Phi = \alpha^{-1}(\Phi^{\alpha})$ is an invariant of B.*

Proof. By Lemma 5, we show that the Algorithm 2 only makes states partition and then has no effect on the simulation relation. If $(l_1, x) \xrightarrow{p} (l, x')$ and $\alpha^{-1}(\phi_1^{\alpha}) = \text{at\_}l_1 \wedge \varphi_1$, then there exists ϕ^{α} such that $\alpha^{-1}(\phi^{\alpha}) = \text{at\_}l \wedge \varphi$, $(l, x')R\phi^{\alpha}$, and $\phi_1^{\alpha} \overset{p}{\rightsquigarrow} \phi^{\alpha}$. If $\varphi_e \Rightarrow \varphi$ does not hold, then the abstract state ϕ^{α} cannot be partitioned by φ_e. Otherwise, we split abstract state ϕ^{α} with disjoint two parts $\phi = \alpha(\text{at\_}l \wedge \varphi_e)$ and $\phi' = \alpha(\text{at\_}l \wedge \varphi')$, and we have $\varphi_e \wedge \varphi' =$ false and $\varphi_e \vee \varphi' = \varphi$, so we have either $\phi_1^{\alpha} \overset{p}{\rightsquigarrow} \phi$ or $\phi_1^{\alpha} \overset{p}{\rightsquigarrow} \phi'$. Then, the refined abstract system B_{ref}^{α} simulates B. If Φ^{α} is an invariant of B_{ref}^{α}, then $\alpha^{-1}(\Phi^{\alpha})$ is an invariant of B. □

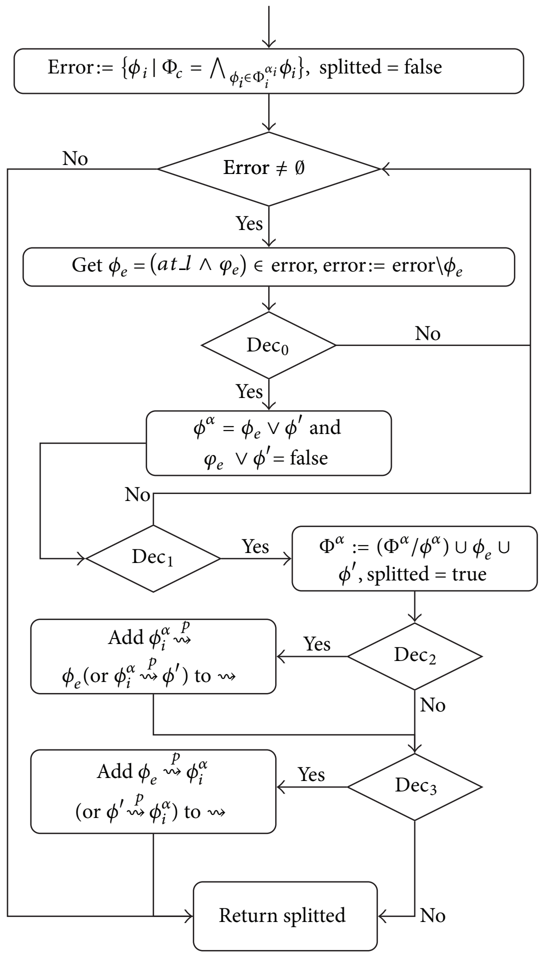
For partition refinement method, given a counterexample $\Phi_c = \bigwedge_{\phi_i \in \Phi_i^{\alpha_i}} \phi_i$, state partition in ith component, based on ϕ_i, only affects one abstract state to be parted. And a state is parted only once by ϕ_i. As exists quantifier for states needs examines all possible abstract state pairs and so does the quantifier of transitions, state partition needs time complexity as $\mathcal{O}(\sum_{i=1}^{n}(|\Phi_i^{\alpha}| \cdot |\rightsquigarrow_i|^2))$.

The above refinement approaches are applied on the parallel compositional system. However, extended with the decidable theories and symbolic representation, our approaches can be applied on a more complex system like the hierarchical component system.

5. Iteration Verification Framework

In this section, we present the unified verification framework composed of compositional abstraction and our proposed refinement techniques.

Our verification framework for component-based systems is an iterative scheme presented by Algorithm 1. In our framework, there exists the method of abstraction refinement



Error:= $\{\phi_i \mid \Phi_c = \bigwedge_{\phi_i \in \Phi_i^{\alpha_i}} \phi_i\}$, splitted = false
Error ≠ ∅
No
Yes
Get $\phi_e = (at\_l \wedge \varphi_e) \in$ error, error:= error$\backslash\phi_e$
Dec$_0$
No
Yes
$\phi^\alpha = \phi_e \vee \phi'$ and $\varphi_e \vee \phi' =$ false
No
Dec$_1$
Yes
$\Phi^\alpha := (\Phi^\alpha/\phi^\alpha) \cup \phi_e \cup \phi'$, splitted = true
Add $\phi_i^\alpha \overset{p}{\rightsquigarrow} \phi_e$(or $\phi_i^\alpha \overset{p}{\rightsquigarrow} \phi'$) to $\rightsquigarrow$
Yes
Dec$_2$
No
Add $\phi_e \overset{p}{\rightsquigarrow} \phi_i^\alpha$ (or $\phi' \overset{p}{\rightsquigarrow} \phi_i^\alpha$) to $\rightsquigarrow$
Yes
Dec$_3$
Return splitted
No

Figure 1

presented by Algorithm 2, which includes the counterexample guided invariant strengthening and abstract state partitioning inspired by CEGAR mechanism. In the Algorithm 1, the unified compositional abstraction refinement for the invariant checking problem on component-based systems is given, which contains two phases.

At the verification phase, we apply compositional verification to the invariant checking problem. If properties can be proven under the current abstraction (see line 12 in Algorithm 1), the verification succeeds immediately. Otherwise, the refinement is performed (see line 16 in Algorithm 1). If the abstraction cannot be further refined by the current reachable counterexample (see line 17 in Algorithm 1), then current counterexample is a genuine counterexample. If current counterexamples are all spurious counterexamples (see line 20 in Algorithm 1), we conclude that verification succeeds. Otherwise, we go to line 23; we get refined abstract system and compute the new more concrete interaction invariant with conjunction of strengthened inductive invariant computed from backward propagation of spurious counterexamples and go back to counterexample guiding abstraction refinement, portion of verification basis, which call for less efforts than previous iteration processes in [11, 12].

At the refinement phase, we first strengthen the interaction invariant and remove the spurious counterexamples. After the invariant is strengthened, we analyze the remaining counterexamples and partition abstract states. When our algorithm returns to the verification phase, added states will induce a refined abstract model. Our refinement method (see Algorithm 2) reconsiders information of counterexamples, which may include $\mathrm{Err}_{\mathrm{I}}$ and $\mathrm{Err}_{\mathrm{II}}$, where $\mathrm{Err}_{\mathrm{I}}$ is unreachable error state introduced by interaction interference and $\mathrm{Err}_{\mathrm{II}}$ is inaccurate local configuration caused by abstraction function. We use the invariant strengthening technique to remove $\mathrm{Err}_{\mathrm{I}}$ from the counterexamples and strengthen the invariant use of the counterexample in $\mathrm{Err}_{\mathrm{I}}$. We use the counterexample in $\mathrm{Err}_{\mathrm{II}}$ to do the state partition refinement which refines the abstraction until a genuine counterexample is found or the safety property holds on abstract systems.

Theorem 11. *Let $\mathcal{S} = \langle \gamma(B_1, \ldots, B_n), Init\rangle$ be a compound component-based system, and let Φ be a specific property predicate.*

(1) *If $IterationVerify(\mathcal{S}, \Phi)$ returns true, then $\mathcal{S} \models \Phi$;*

(2) *if $IterationVerify(\mathcal{S}, \Phi)$ returns false, then $\mathcal{S} \not\models \Phi$.*

```
     Input: Component system $\mathcal{S} = (\gamma(B_1, \ldots, B_n), Init)$, property $\Phi$
     Output: $true$ or $false$
(1)  $\Phi_i = true$ for each $i = 1, \ldots, n$, $Refinable := true$
(2)  for $i \leftarrow 1$ to $n$ do
(3)      compute the component invariant $\Phi_i'$ based on $B_i$
(4)      $\Phi_i := \Phi_i \wedge \Phi_i'$
(5)      compute the corresponding abstraction $B_i^{\alpha_i}$ based on $B_i$ and $\Phi_i$
(6)  end
(7)  from $\gamma(B_1^{\alpha_1}, \ldots, B_n^{\alpha_n})$, compute $L_1, L_2, \ldots, L_m$
(8)  for $k \leftarrow 1$ to $m$ do
(9)      $\Psi_k = \bigvee_{\phi \in L_k} \alpha^{-1}(\phi)$
(10) end
(11) $\Psi := \bigwedge_{k=1}^{m} \Psi_k$
(12) if $\neg\Phi \wedge \Psi \wedge (\bigwedge_{i=1}^{n} \Phi_i) = false$ then
(13)     $\Phi$ is an invariant of $\mathcal{S}$
(14)     return $true$
(15) else
(16)     compute counterexample set $Err := \{\Phi_c\}$
         $Refinable := AbsRefinement(\mathcal{S}^{\alpha}, Err, \Psi)$
(17)     if $Refinable = false$&&$Err \neq \emptyset$ then
(18)         return $false$
(19)     end
(20)     if $Err = \emptyset$ then
(21)         return $true$
(22)     else
(23)         goto 12
(24)     end
(25) end
```

Algorithm 1: Iteration verification.

Proof. (1) If Algorithm 1 returns true from line 14, with the precondition $\neg\Phi \wedge \Psi \wedge (\bigwedge_{i=1}^{n} \Psi_i) = \text{false}$, we get the conclusion that the property satisfied immediately. Otherwise, if all current counterexamples are spurious counterexamples, the verification stops with the property satisfied at line 21. (2) If Algorithm 1 returns false from line 18 with Refinable = false and Err $\neq \emptyset$, we get that the abstract system is unable to be refined any more by the current counterexample, which means the counterexample is a real violated behavior truly reachable from the initial configuration and cannot be eliminated by a more concrete system. Otherwise, abstract system should be refined by the next refinement using the current counterexamples. □

Theorem 12. *Let $\mathcal{S} = \langle \gamma(B_1, \ldots, B_n), Init \rangle$ be a compound component-based system, and let Φ be a specific property predicate. Algorithm $IterationVerify(\mathcal{S}, \Phi)$ always terminates.*

Proof. In each refinement iteration, either some new strengthening invariant is added or the abstract system state is split into new states. Note that the state space of abstract system is finite and that the number of possible split states is also finite. In the worst case, the algorithm finally terminated with the refined system is just the original system. Then the algorithm can terminate finally. □

Theorem 13. *Let $\mathcal{S} = \langle \gamma(B_1, \ldots, B_n), Init \rangle$ be a compound component-based system, and let Φ be a specific property predicate.*

(1) *If $\mathcal{S} \models \Phi$, then $IterationVerify(\mathcal{S}, \Phi)$ returns true;*

(2) *if $\mathcal{S} \not\models \Phi$, then $IterationVerify(\mathcal{S}, \Phi)$ returns $false$.*

Proof. (1) According to the second statements of Theorem 11, if $\mathcal{S} \models \Phi$, Algorithm 1 cannot return with false. According to Theorem 12, Algorithm 1 always terminates. Thus, if $\mathcal{S} \models \Phi$, then algorithm can only terminate with true.

(2) Similarly, according to the first statement of Theorems 11 and 12, if $\mathcal{S} \not\models \Phi$, the algorithm can only terminate with false. □

6. Examples and Experiments

We will provide certain examples to illustrate the effectiveness of our proposed verification framework.

6.1. Train Gate Controller. Consider the example of the train gate controller [18], which consists of a *controller*, a *gate*, and a number of *trains*. The model presented in Figure 2 describes only one train interacting with the controller and the gate. The controller operates the gate up and down when a train enters and exits, respectively.

The *Controller* has four locations $\{c_0, c_1, c_2, c_3\}$, one variable z, five ports {approach, raise, exit, lower, tick}, and eight guarded transitions. The *Train* has three locations {far, in, near}, a variable x, four ports {approach, exit, t, tick}, and six guarded transitions. The *Gate* has four locations $\{g_0, g_1, g_2, g_3\}$, a variable

```
Input: Abstract system $\mathcal{S}^\alpha = (\gamma(B_1^{\alpha_1}, \dots, B_n^{\alpha_n}), Init^\alpha)$,
       set of counterexamples $Err$, interaction invariant $\Psi$
Output: $true$ or $false$
(1)  $Refined := false, Err_I := \emptyset, Err_{II} := \emptyset$
(2)  foreach $\Phi_c \in Err$ do
(3)      $Err := Err \setminus \Phi_c$
(4)      $InvStr_{\Phi_c} := InvarStrengthen(\mathcal{S}^\alpha, \Phi_c)$
(5)      if $InvStr_{\Phi_c} = true$ then
(6)          $Err_{II} := Err_{II} \cup \Phi_c$
(7)      else
(8)          $Err_I := Err_I \cup \Phi_c$
(9)      end
(10) end
(11) $Err := Err_{II}$
(12) if $Err_{II} = \emptyset$ then
(13)     return $false$
(14) else
(15)     foreach $\Phi_c \in Err_{II}$ do
(16)         if $SplitAbstraction(\mathcal{S}^\alpha, \Phi_c) = true$ then
(17)             $Refined := true$
(18)         end
(19)     end
(20)     from split refinement abstract system $\mathcal{S}^\alpha_{ref}$, compute
         new compute interaction invariant $\Psi$
(21)     foreach $\Phi_c \in Err_I$ do
(22)         $\Psi := \Psi \wedge InvStr_{\Phi_c}$
(23)     end
(24)     return $Refined$
(25) end
```

Algorithm 2: Abstraction refinement.

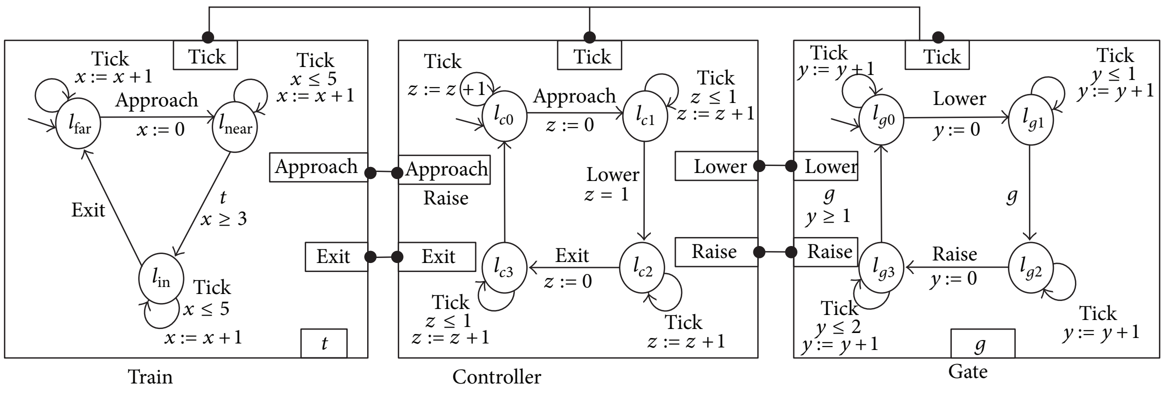


Figure 2: The component-based model for train gate controller (TGC).

y, four ports {lower, raise, g, tick}, and eight guarded transitions. Three atomic components Train, Gate, and Controller are composed with a set γ of interactions: {C.tick, T.tick, G.tick}, {T.approach, C.approach}, {T.exit, C.exit}, {C.lower, G.lower}, and {C.raise, G.raise}. In this example, we aim to verify the TGC system by the initial condition Init $= l_{far} \wedge (x = 3) \wedge l_{c_0} \wedge (z = 1) \wedge l_{g_0} \wedge (y = 1)$. For the above three atomic components, we generate the following predicates $\Phi_{Controller} = (l_{c_0} \wedge z \ge 0) \vee (l_{c_1} \wedge 0 \le z \le 1) \vee (l_{c_2} \wedge z \ge 1) \vee (l_{c_3} \wedge 0 \le z \le 1)$, $\Phi_{Train} = (l_{far} \wedge x \ge 3) \vee (l_{near} \wedge 0 \le x \le 5) \vee (l_{in} \wedge 3 \le x \le 5)$, and $\Phi_{Gate} = (l_{g_0} \wedge y \ge 1) \vee (l_{g_1} \wedge 0 \le y \le 1) \vee (l_{g_2} \wedge y \ge 0) \vee (l_{g_3} \wedge 0 \le y \le 2)$, which are, respectively, the component invariants of Train, Gate, and Controller. Since this system has an infinite number of reachable states. With the application of abstraction function α, we transform the original components (infinite) to the computed abstract components (finite) presented in Figure 3. The computed abstract states are $\{\phi_{n_1}, \phi_{n_2}, \phi_{n_3}, \phi_{in}, \phi_{far}, \phi_{c_{11}}, \phi_{c_{12}}, \phi_{c_2}, \phi_{c_{31}}, \phi_{c_{32}}, \phi_{c_0}, \phi_{g_{11}}, \phi_{g_{12}}, \phi_{g_2}, \phi_{g_{31}}, \phi_{g_{32}}, \phi_{g_0}\}$ (see Table 1).

Table 1

Train$^\alpha$	Controller$^\alpha$	Gate$^\alpha$
$\phi_{n_1} = l_{\text{near}} \wedge x = 0$	$\phi_{c_{11}} = l_{c_1} \wedge z = 0$	$\phi_{g_{11}} = l_{g_1} \wedge y = 0$
$\phi_{n_2} = l_{\text{near}} \wedge 1 \leq x \leq 2$	$\phi_{c_{12}} = l_{c_1} \wedge z = 1$	$\phi_{g_{12}} = l_{g_1} \wedge y = 1$
$\phi_{n_3} = l_{\text{near}} \wedge 3 \leq x \leq 5$	$\phi_{c_2} = l_{c_2} \wedge z \geq 1$	$\phi_{g_2} = l_{g_2} \wedge y \geq 0$
$\phi_{\text{in}} = l_{\text{in}} \wedge 3 \leq x \leq 5$	$\phi_{c_{31}} = l_{c_3} \wedge z = 0$	$\phi_{g_{31}} = l_{g_3} \wedge y = 0$
$\phi_{\text{far}} = l_{\text{far}} \wedge x \geq 3$	$\phi_{c_{32}} = l_{c_3} \wedge z = 1$	$\phi_{g_{32}} = l_{g_3} \wedge 1 \leq y$
	$\phi_{c_0} = l_{c_0} \wedge z \geq 0$	$\phi_{g_0} = l_{g_0} \wedge y \geq 1$

The interested safety property is that when the train is at the location l_{in}, the gate is at l_{g_2}. We apply the compositional verification rule to check the property. In this model, the verification rule can infer this property, so we get the conclusion of the properties satisfied.

6.2. Temperature Control System. Consider the example of the temperature control system [11, 12] presented in Figure 4. The model consists of three atomic components: Controller, Rod$_1$, and Rod$_2$.

The Controller has two locations $\{l_5, l_6\}$, a variable θ, three ports {tick, cool, heat}, and four guarded transitions. The Rod$_1$ has two locations $\{l_1, l_2\}$, a variable t_1, three ports {tick$_1$, cool$_1$, rest$_1$}, and four guarded transitions. Likewise, the Rod$_2$ has two locations $\{l_3, l_4\}$, a variable t_2, three ports {tick$_2$, cool$_2$, rest$_2$}, and four guarded transitions. Three atomic components Controller, Rod$_1$, and Rod$_2$ are composed with a set γ of interactions: {tick, tick$_1$, tick$_2$}, {cool, cool$_1$}, {cool, cool$_2$}, {heat, rest$_1$}, and {heat, rest$_2$}. In this example, we aim to verify deadlock-freedom of the temperature control system by the initial condition Init $= l_5 \wedge (\theta = 100) \wedge l_1 \wedge (t_1 = 3600) \wedge l_3 \wedge (t_2 = 3600)$.

For the above three atomic components, we generate the following predicates $\Phi_1 = (l_1 \wedge t_1 \geq 0) \vee (l_2 \wedge t_1 \geq 3600)$, $\Phi_2 = (l_3 \wedge t_2 \geq 0) \vee (l_4 \wedge t_1 \geq 3600)$, and $\Phi_3 = (l_5 \wedge 100 \leq \theta \leq 1000) \vee (l_6 \wedge 100 \leq \theta \leq 1000)$, which are, respectively, the component invariants of Rod$_1$, Rod$_2$, and Controller. Since this system has an infinite number of reachable states, with the application of abstraction function α, we transform the original components (infinite) to the computed abstract components (finite) in Figure 5. The computed abstract states are $\{\phi_{11}, \phi_{12}, \phi_{21}, \phi_{22}, \phi_{31}, \phi_{32}, \phi_{41}, \phi_{42}, \phi_{51}, \phi_{52}, \phi_{61}, \phi_{62}\}$.

The interaction invariant of the component system is generated from the concept of the trap. The sets of traps for the abstract system are $L_1 = \{\phi_{21}, \phi_{41}, \phi_{51}, \phi_{52}\}$, $L_2 = \{\phi_{11}, \phi_{12}, \phi_{21}, \phi_{31}, \phi_{32}, \phi_{41}\}$, $L_3 = \{\phi_{32}, \phi_{41}, \phi_{42}, \phi_{51}\}$, $L_4 = \{\phi_{11}, \phi_{12}, \phi_{31}, \phi_{32}, \phi_{61}, \phi_{62}\}$, and $L_5 = \{\phi_{12}, \phi_{21}, \phi_{22}, \phi_{51}\}$. After the computation of traps, we generate interaction invariants $\Psi_i = \bigvee_{\phi \in L_i} \alpha^{-1}(\phi)$ $(i = 1, 5)$ and then get $\Psi := \bigwedge_{i=1}^{5} \Psi_i$.

To verify the deadlock-freedom of the temperature control system, we define a predicate *DIS* which characterizes the set of system states from which all interactions in γ are disabled. In this example, DIS $= (\neg(l_5 \wedge \theta < 1000)) \bigwedge (\neg(l_6 \wedge \theta = 100) \vee \neg l_2) \bigwedge (\neg(l_6 \wedge \theta > 100)) \bigwedge (\neg(l_5 \wedge \theta = 1000) \vee \neg(l_3 \wedge t_2 \geq 3600)) \bigwedge (\neg(l_5 \wedge \theta = 1000) \vee \neg(l_1 \wedge t_1 \geq 3600)) \bigwedge (\neg(_6 \wedge \theta = 100) \vee \neg l_4)$. If the predicate $\neg$DIS is an invariant of the temperature control system, then it is deadlock-free. To check that $\neg$DIS is an invariant, we need a stronger invariant Φ such that $\Phi \Rightarrow \neg$DIS (equivalently, $\Phi \wedge$ DIS $=$ false). Based on the previous compositional verification rule, the computed invariant is $\Phi = (\Phi_1 \wedge \Phi_2 \wedge \Phi_3) \wedge \Psi$. To verify the deadlock-freedom of the system, we computed the predicate $\Phi \wedge$ Init $\wedge$ DIS, which is reduced to

(1) $\Phi_{c1} = (l_1 \wedge 1 \leq t_1 < 3600) \wedge (l_3 \wedge 1 \leq t_2 < 3600) \wedge (l_5 \wedge \theta = 1000)$;

(2) $\Phi_{c2} = (l_1 \wedge 1 \leq t_1 < 3600) \wedge (l_4 \wedge t_2 \geq 3600) \wedge (l_5 \wedge \theta = 1000)$;

(3) $\Phi_{c3} = (l_2 \wedge t_1 \geq 3600) \wedge (l_3 \wedge 1 \leq t_2 < 3600) \wedge (l_5 \wedge \theta = 1000)$.

As the invariant $\Phi = (\Phi_1 \wedge \Phi_2 \wedge \Phi_3) \wedge \Psi$ is the *overapproximate* of the reachable states, some spurious counterexamples may be included. We can strengthen invariants and refine the abstract system by our proposed refinement approaches. At first, we analyze the generated counterexamples Φ_{c1}, Φ_{c2}, and Φ_{c3} and check whether the counterexamples are spurious or not by our counterexample-guided invariant strengthening. The abstract states $\Phi_2 = \phi_{12} \wedge \phi_{42} \wedge \phi_{52}$ and $\Phi_3 = \phi_{21} \wedge \phi_{32} \wedge \phi_{52}$, which include Φ_{c2} and Φ_{c3}, are unreachable from the initial abstract states. Since Φ_2 and Φ_3 are unreachable from the initial abstract states, we can conclude that Φ_{c2} and Φ_{c3} are spurious counterexamples. However, we cannot make decisions whether Φ_{c1} is reachable from the initial states, although Φ_1 is reachable from the initial abstract states.

After the application of invariant strengthening on the abstract system, we eliminate Φ_{c2} and Φ_{c3} from counterexamples and strengthen invariants by the generated infeasible states. Now, we refine the abstract system and check whether Φ_{c1} is genuine counterexample or not by state partition technique. After the application of state partition technique on the abstract system, we get the refined abstract system presented in Figure 6. We split the state $\phi_{12} = l_1 \wedge t_1 \geq 1$ into $\phi_{121} = l_1 \wedge 1 \leq t_1 < 3600$ and $\phi_{122} = l_1 \wedge t_1 \geq 3600$; state $\phi_{32} = l_3 \wedge t_2 \geq 1$ into $\phi_{321} = l_3 \wedge 1 \leq t_2 < 3600$ and $\phi_{322} = l_3 \wedge t_2 \geq 3600$; state $\phi_{52} = l_5 \wedge 101 \leq \theta \leq 1000$ into $\phi_{521} = l_5 \wedge 101 \leq \theta < 1000$ and $\phi_{522} = l_5 \wedge \theta = 1000$. By the state partition technique, finally we find that Φ_{c1} is a counterexample of the system.

The example is implemented as BIP models and has been translated into timed automata using the tool BIP2UPPAAL [19]. To illustrate effectiveness of our approach, we verify models against property EFΦ_{c1}, EFΦ_{c2}, and EFΦ_{c3}, which means "there exists a run that eventually reaches deadlock states Φ_{ci}." As a comparison, we first check both original system and abstract system. The checking results are shown in Table 2. As EFΦ_{c2} and EFΦ_{c3} are not satisfied by the abstract model, we conclude that Φ_{c2} and Φ_{c3} are spurious counterexamples immediately. Then we make partition refinement with Φ_{c2}. During state partition refinement, states $P12$, $P52$, and $P32$ (refer to ϕ_{12}, ϕ_{52}, and ϕ_{32}, resp.) have been partitioned. We verified refined abstract models against property. As EFΦ_{c1} is ultimately satisfied and we cannot use Φ_{c1} to partition models further, we conclude that Φ_{c1} is a genuine counterexample.

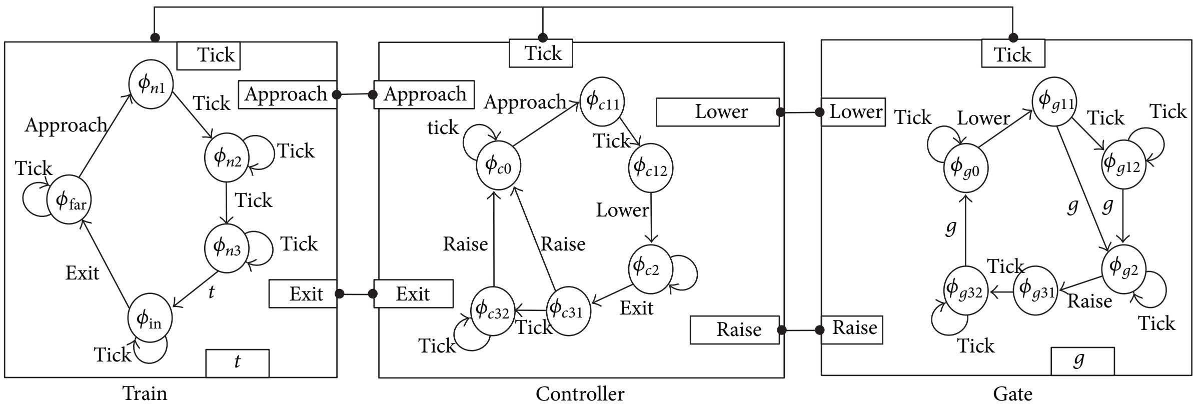


FIGURE 3: The abstraction for the TGC.

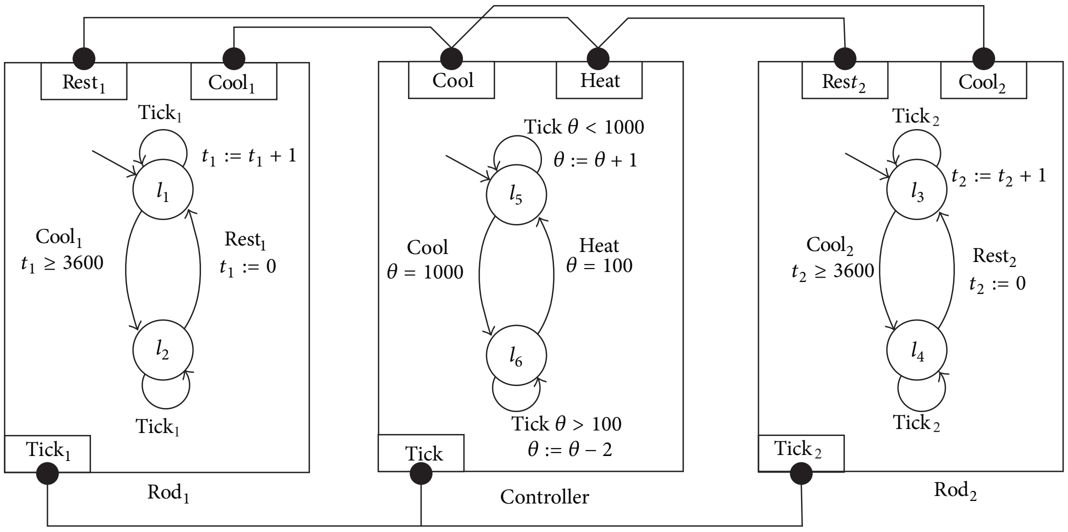


FIGURE 4: The component-based model for temperature control system.

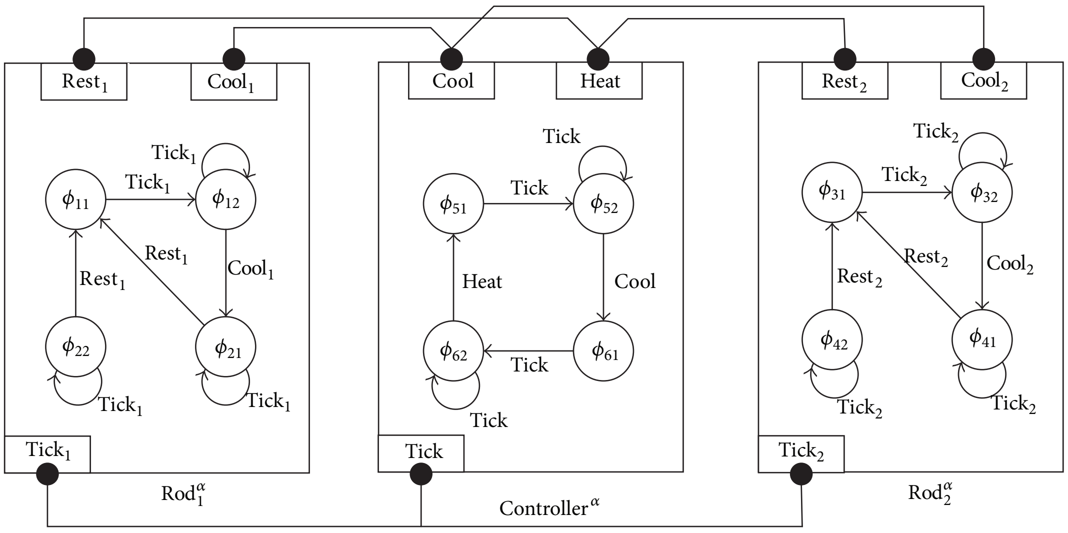


FIGURE 5: The abstraction for the component-based model.

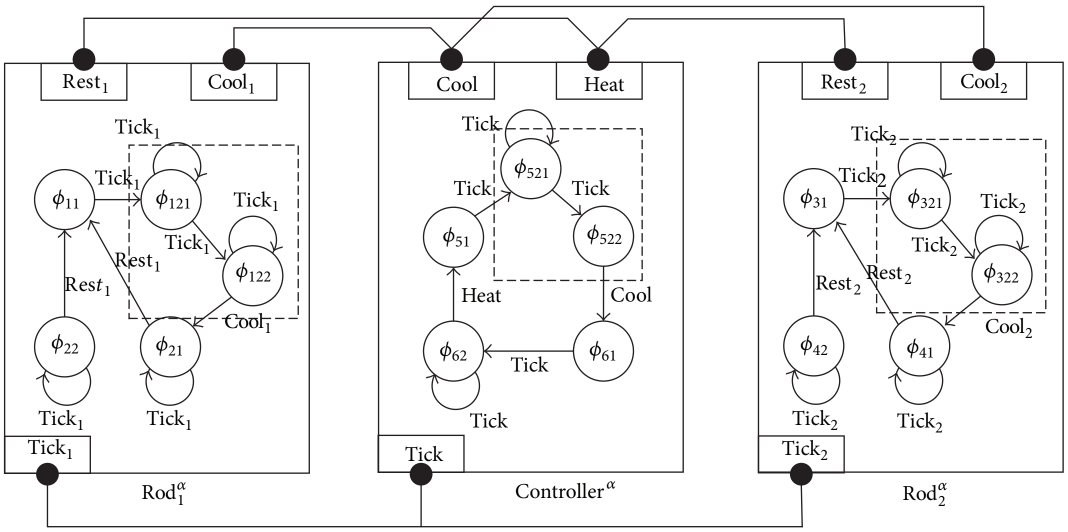


Figure 6: The refined abstract model.

Table 2: Simple table.

	Original	Abstract	Refined
Property_1		$\mathrm{EF}\Phi_{c1}$	
Result	True	True	True
Time (s)	0.046	0.001	0.001
Memory (KB)	8,056	7,084	7,084
Property_2		$\mathrm{EF}\Phi_{c2}$	
Result	False	False	False
Time (s)	0.044	0.001	0.001
Memory (KB)	8,060	7,088	7,072
Property_3		$\mathrm{EF}\Phi_{c3}$	
Result	False	False	False
Time (s)	0.045	0.001	0.001
Memory (KB)	8,064	7,084	7,072

From the above analysis, we have shown the effectiveness of our proposed verification framework. Using our proposed refinement techniques, we reconsidered the counterexamples of the system. We finally identify which are spurious counterexamples and refine the abstract systems.

7. Conclusions

We have proposed a unified framework with iterative refinements for compositional verification of component-based systems. This framework extends invariant-based compositional verification rule with the counterexample-guided invariant strength and state partition techniques. The former removes the spurious counterexample and uses the fixed point generated backward from the spurious to strengthen the system invariant. The latter partitions the abstract states according to the rest counterexamples to refine the abstract system. Both contribute to the unified verification framework which is proved to be sound and complete.

Compared with the invariant-based compositional verification which is incomplete, our verification framework with the iterative refinements can get more precise results with the balance of the verification complexity.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

This work was supported by the Fund NSFC61171121 and the Science Foundation of Chinese Ministry of Education-China Mobile 2012.

References

[1] K. M. Chandy, *Parallel Program Design*, Springer, 1989.

[2] S. Bensalem, Y. Lakhnech, and S. Owre, “Computing abstractions of infinite state systems compositionally and automatically,” in *Computer Aided Verification*, pp. 319–331, Springer, 1998.

[3] S. Chaki, J. Ouaknine, K. Yorav, and E. Clarke, “Automated compositional abstraction refinement for concurrent C programs: a two-level approach,” *Electronic Notes in Theoretical Computer Science*, vol. 89, no. 3, pp. 417–432, 2003.

[4] C. Flanagan and S. Qadeer, “Thread-modular model checking,” in *Model Checking Software*, pp. 213–224, Springer, 2003.

[5] A. Malkis, A. Podelski, and A. Rybalchenko, “Thread-modular counterexample-guided abstraction refinement,” in *Static Analysis*, pp. 356–372, Springer, 2011.

[6] T. A. Henzinger, R. Jhala, R. Majumdar, and S. Qadeer, “Thread-modular abstraction refinement,” in *Computer Aided Verification*, pp. 262–274, Springer, Berlin, 2003.

[7] T. A. Henzinger, R. Jhala, and R. Majumdar, “Race checking by context inference,” *ACM SIGPLAN Notices*, vol. 39, no. 6, pp. 1–13, 2004.

[8] S. Bensalem, Y. Lakhnech, and S. Owre, “Invest: a tool for the verification of invariants,” in *Computer Aided Verification*, pp. 505–510, Springer, 1998.

[9] A. Pnueli, S. Ruah, and L. Zuck, “Automatic deductive verification with invisible invariants,” in *Tools and Algorithms for the Construction and Analysis of Systems*, pp. 82–97, Springer, 2001.

[10] T. Arons, A. Pnueli, S. Ruah, Y. Xu, and L. Zuck, “Parameterized verification with automatically computed inductive assertions,” in *Computer Aided Verification*, pp. 221–234, Springer, Berlin, 2001.

[11] S. Bensalem, M. Bozga, T.-H. Nguyen et al., “Compositional verification for component-based systems and application,” in *Proceedings of the Automated Technology for Verification and Analysis 6th International Symposium (ATVA ’08)*, vol. 5311, pp. 64–79, Seoul, Republic of Korea, 2008.

[12] S. Bensalem, M. Bozga, T.-H. Nguyen, and J. Sifakis, “Compositional verification for component-based systems and application,” *IET Software*, vol. 4, no. 3, pp. 181–193, 2010.

[13] M. Bozga, M. Jaber, and J. Sifakis, “Source-to-source architecture transformation for performance optimization in BIP,” *IEEE Transactions on Industrial Informatics*, vol. 6, no. 4, pp. 708–718, 2010.

[14] S. Bensalem and Y. Lakhnech, “Automatic generation of invariants,” *Formal Methods in System Design*, vol. 15, no. 1, pp. 75–92, 1999.

[15] J. L. Peterson, *Petri Net Theory and the Modeling of Systems*, Prentice Hall, Englewood Cliffs, NJ, USA, 1981.

[16] K. Barkaoui and M. Minoux, “A polynomial-time graph algorithm to decide liveness of some basic classes of bounded Petri nets,” in *Application and Theory of Petri Nets*, vol. 616, pp. 62–75, Springer, Berlin, 1992.

[17] A. Tiwari, H. Rueß, H. Saïdi, and N. Shankar, “A technique for invariant generation,” in *Tools and Algorithms for the Construction and Analysis of Systems*, pp. 113–127, Springer, 2001.

[18] R. Alur and D. L. Dill, “A theory of timed automata,” *Theoretical Computer Science*, vol. 126, no. 2, pp. 183–235, 1994.

[19] C. Su, M. Zhou, L. Yin, H. Wan, and M. Gu, “Modeling and verification of component-based systems with data passing using bip,” in *Proceedings of the 18th IEEE International Conference on Engineering of Complex Computer Systems (ICECCS ’13)*, pp. 4–13, 2013.

Hindawi Publishing Corporation
Journal of Applied Mathematics
Volume 2014, Article ID 194574, 15 pages
http://dx.doi.org/10.1155/2014/194574

Research Article

Groebner Bases Based Verification Solution for SystemVerilog Concurrent Assertions

Ning Zhou,[1,2] Xinyan Gao,[3] Jinzhao Wu,[1,4] Jianchao Wei,[3] and Dakui Li[3]

[1] *School of Computer and Information Technology, Beijing Jiaotong University, Beijing 10044, China*
[2] *School of Electronic and Information Engineering, Lanzhou Jiaotong University, Lanzhou 730070, China*
[3] *G & S Labs, School of Software of Dalian University of Technology, Dalian 116620, China*
[4] *Guangxi Key Laboratory of Hybrid Computation and IC Design Analysis, Guangxi University for Nationalities, Nanning 530006, China*

Correspondence should be addressed to Jinzhao Wu; jzwu205@gmail.com

Received 13 February 2014; Accepted 7 April 2014; Published 11 June 2014

Academic Editor: Xiaoyu Song

We introduce an approach exploiting the power of polynomial ring algebra to perform SystemVerilog assertion verification over digital circuit systems. This method is based on Groebner bases theory and sequential properties checking. We define a constrained subset of SVAs so that an efficient polynomial modeling mechanism for both circuit descriptions and assertions can be applied. We present an algorithm framework based on the algebraic representations using Groebner bases for concurrent SVAs checking. Case studies show that computer algebra can provide canonical symbolic representations for both assertions and circuit designs and can act as a novel solver engine from the viewpoint of symbolic computation.

1. Introduction

SystemVerilog [1, 2] is the most important unified Hardware Description and Verification Language (HDVL) and provides a major set of extensions from the Verilog language with the added benefit of supporting object orientated constructs and assertions feature. SystemVerilog provides special language constructs and assertions, to specify and verify design behavior. An assertion is a statement that a specific condition, or sequence of conditions, in a design is true. In the industry of integrated circuits design, Assertions-based verification (ABV) using SystemVerilog Assertions (SVAs) is now changing the traditional design process. With the help of SVAs, it is easy to formally characterize the design requirements at various levels of abstraction, guide the verification task, and simplify the design of the testbench. Assertions essentially become active design comments, and one important methodology treats them exactly like active design comments.

Moreover, assertions can be attached directly to RTL, working in cycle-precise domain, or they can operate in transactional domain with the help of monitors data extractors. Then, these modular checkers minimize or even eliminate the need of model checkers, which consist of HDL modules to provide the verification.

An important benefit of assertions is the ease of specifying functional coverage. Simulation tools compute the functional coverage, as defined by the assertions, which add a greater level of assurance that the testbench invoked the desired functions.

More recently, the key EDA tool vendors have researched new verification methodologies and languages, which implement functional code coverage, assertion-based coverage. and constrained random techniques inside a complete verification environment.

In [3], a subset of SystemVerilog assertions is defined to apply induction-based bounded model checking (BMC) to this subset of SVAs within acceptable run times and moderate memory requirements. As is well known, the conventional simulation for assertion checking is the well-understood and most commonly used technique, but only feasible for very small scale systems and cannot provide exhaustive checking. While symbolic simulation proposed by Darringer [4] as early as 1979 can provide exhaustive checking by covering many conditions with a single simulation sequence, but it could not handle large circuits due to exponential symbolic expressions. Earlier work in applications of symbolic manipulation and algebraic computation has gained significant extensions and improvements. In [5], a technique framework on Groebner bases demonstrated that computer algebraic geometry method can be used to perform symbolic model checking using an encoding of Boolean sets as the common zeros of sets of polynomials. In [6], a similar technique to framework based Wu's Method has been further extended to bit level symbolic model checking. In [7], an improved framework of multivalued model checking via Groebner bases method was proposed, which is based on a canonical polynomial representation of the multivalued logics.

In our previous work [8], we proposed a method using Groebner bases to perform SEREs assertion verification for synchronous digital circuit systems. Then we introduced a verification solution based on Wu's method towards SystemVerilog assertion checking [9]. This paper aims to verify whether a SVA property holds or not on the traces produced after several cycles running over a given synchronous sequential circuit. It is the follow-up work of [9]. Groebner Bases and Wu's method are the most important methods of computer algebra. Wu's method of characteristic set is a powerful theorem proving technique, and Groebner Bases allows many important properties of the ideal and the associated algebraic variety to be deduced easily. So Groebner Bases method has a better theoretical guide, and the algorithm based on Wu's method is more efficient.

Checking SVAs is computationally very complex in general while for practical purposes a subset is sufficient. In this work we

(1) define a constrained subset of SVAs;
(2) perform algebraization of SVA operators for the constrained subset;
(3) do translation of SVAs into polynomial set representations;
(4) provide a symbolic computation based algebraic algorithm for SVAs verification.

Our approach can handle safety properties that argue over a bounded number of time steps. Local variables in SVAs can be handled as symbolic constant without any temporal information. Nevertheless, liveness properties and infinite sequences in SVAs are excluded.

2. Preliminaries

In this section, to keep the paper self-contained, we will give the basics of SystemVerilog and algebraic symbolic computation used throughout this paper.

2.1. SystemVerilog Preliminary. SystemVerilog is an IEEE-approved (IEEE 1800-2005) [1] hardware description language. It provides superior capabilities for system architecture, design, and verification.

SystemVerilog has combined many of the best features of both VHDL and Verilog.

Therefore, on the one hand, VHDL users will recognize many of the SystemVerilog constructs, such as enumerated types, records, and multidimensional arrays. On the other hand, Verilog users can reuse existing designs; SystemVerilog is a superset of Verilog so no modification of existing Verilog code is required.

Generally, the SystemVerilog language provides three important benefits over Verilog.

(1) *Explicit design intent*—SystemVerilog introduces several constructs that allow designers to explicitly state what type of logic should be generated.
(2) *Conciseness of expressions*—SystemVerilog includes commands that allow the users to specify design behavior more concisely than previously possible.
(3) *High level design abstraction*—The SystemVerilog interface construct facilitates intermodule communication.

Especially, SystemVerilog provides special language constructs and assertions, to verify design behavior. An assertion is a statement that a specific condition, or sequence of conditions, in a design is true. If the condition or sequence is not true, the assertion statement will generate an error message.

Additionally, one important capability in SystemVerilog is the ability to define assertions outside of Verilog modules and then bind them to a specific module or module instance. This feature allows test engineers to add assertions to existing Verilog models, without having to change the model in any way. One of the goals of SystemVerilog assertions is to provide a common semantic meaning for assertions so that they can be used to drive various design and verification tools.

In SystemVerilog, there are two types of assertions.

(1) Immediate Assertions. Immediate assertions follow simulation event semantics for their execution and are executed like a statement in a procedural block. Immediate assertions are primarily intended to be used with simulation and evaluate using simulation event-based semantics.

(2) Concurrent Assertions. Concurrent assertions are based on clock semantics and use sampled values of variables. All timing glitches (real or artificial due to delay modeling and transient behavior within the simulator) are abstracted away. Concurrent assertions can be used in always block or initial block as a statement, or a module as a concurrent block, or an interface block as a concurrent block, or a program block as a concurrent block.

An example of a property using sequence and formal argument is shown below.

```
property test [(req, c, ack)];

    @(posedge clk)
    req |- > c ##1 ack;

endproperty [: test].
```

This property states that signal *req* and then signal *c* become high and signal *ack* will be high in the next cycle.

As illustrated in this example, a concurrent assertion property in SystemVerilog will never be evaluated by itself except when it is invoked by a verification statement. Therefore, the statement: **assert property** $test(a, b, c)$ will cause the checker to perform assertion checking.

Basically, the verification statement in SVA has three forms described as follows:

(i) **assert** to specify the property as a checker to ensure that the property holds for the design;

(ii) **assume** to specify the property as an assumption for the environment. Simulators check that the property holds, while formal tools use the information to generate input stimulus. The purpose of the assume statement is to allow properties to be considered as assumptions for formal analysis as well as for dynamic simulation tools;

(iii) **cover** to monitor the property evaluation for coverage.

When a property is assumed, the tools constrain the environment so that the property holds. In simulation, asserted and assumed properties are continuously verified to ensure that the design or the testbench never violate them.

In some tools, the assumptions on the environment can be used as sequential constraints on the DUT (device under test) inputs in constrained-random simulation.

These proofs are usually subject to other properties that describe the assumed behavior of the environment using assume property statements.

2.2. Groebner Bases Preliminary. Firstly, we will recall some of the key notions of Groebner bases theory and symbolic computation. More detailed and elementary introduction to this subject can be available in books, such as those by Little et al. [10] or those by Becker and Weispfenning [11].

We begin by listing some general facts and establishing notations.

Let k be an algebraically closed field, and let $k[x_1, \ldots, x_n]$ be the polynomial ring in variables $x_1, x_2, \ldots, x_n$ with coefficient in k, under addition and multiplication of polynomial. The basic structure of polynomial rings is given in terms of subsets called ideals which is closed under addition and closed under multiplication by any element of the ring.

Here, let $I \subseteq k[x_1, \ldots, x_n]$ be an ideal. As we all know, the following theorem holds.

Theorem 1 (Hilbert basis theorem). *Every ideal $I \subset k[x_1, \ldots, x_n]$ has a finite generating set. That is, $I = \langle g_1, \ldots, g_t \rangle$ for some $g_1, \ldots, g_t \in I$.*

Then, by the Hilbert basis theorem, there exist finitely many polynomials $f_1, \ldots, f_m$ such that $I = \langle f_1, \ldots, f_m \rangle$. A polynomial $f \subseteq k[x_1, \ldots, x_n]$ defines a map $f : k^n \to k$ via evaluation $(a_1, \ldots, a_n) \mapsto f(a_1, \ldots, a_n)$.

The set $V(I) := a \in k^n \mid \forall f \in I : f(a) = 0 \subseteq k^n$ is called the variety associated with I.

If $V_1 = V(I_1)$ and $V_2 = V(I_2)$ are the varieties defined by ideals I_1 and I_2, then we have $V_1 \cap V_2 = V(\langle I_1, I_2 \rangle)$ and $V_1 \cup V_2 = V(I_1 \times I_2)$, where $I_1 \times I_2 = \langle f_1 f_2 \mid f_1 \in I_1, f_2 \in I_2 \rangle$. If $I_1 = \langle f_1, \ldots, f_r \rangle$ and $I_2 = \langle h_1, \ldots, h_s \rangle$, then $I_1 \times I_2 = \langle f_i \times g_j \mid 1 \le i \le r,\ 1 \le j \le s \rangle$.

Any set of points in k^n can be regarded as the variety of some ideal. Note that there will be more than one ideal defining a given variety. For example, the ideals $\langle x_0 \rangle$ and $\langle x_0, x_1 x_0 - 1 \rangle$ both define the variety $V(x_0)$.

In order to perform verification, we need to be able to determine when two ideals represent the same set of points. That is to say, we need a canonical representation for any ideal. Groebner bases can be used for this purpose.

An essential ingredient for defining Groebner bases is a monomial ordering on a polynomial ring $k[x_1, \ldots, x_n]$, which allows us to pick out a leading term for any polynomial.

Definition 2 (monomial ordering). A monomial ordering on $k[x_1, \ldots, x_n]$ is any relation $\prec$ on $Z^n_{\ge 0}$, or equivalently, any relation on the set of monomials x^α, $\alpha \in Z^n_{\ge 0}$, satisfying

(i) $\prec$ is a total (or linear) ordering on $Z^n_{\ge 0}$;

(ii) $\prec$ is a well-ordering. This means that every nonempty subset of $Z^n_{\ge 0}$ has a smallest element under $\prec$;

(iii) for all $\gamma \in Z^n_{\ge 0}$, $\alpha \prec \beta \Rightarrow \alpha + \gamma \prec \beta + \gamma$.

Examples of monomial ordering include lexicographic order, graded lexicographic order, and graded reverse lexicographic order.

Definition 3 (lexicographic order). Let $\alpha = (\alpha_1, \ldots, \alpha_n)$ and $\beta = (\beta_1, \ldots, \beta_n) \in Z^n_{\ge 0}$. We say $\alpha \prec_{\text{lex}} \beta$ if, in the vector difference $\alpha - \beta \in Z^n$, the leftmost nonzero entry is positive. We will write $x^\alpha \prec_{\text{lex}} x^\beta$ if $\alpha \prec_{\text{lex}} \beta$.

Definition 4 (Groebner basis). Fix a monomial order. A finite subset $G = \{g_1, \ldots, g_t\}$ of an ideal I is said to be a Groebner basis (or standard basis) if $\langle LT(g_1), \ldots, LT(g_t) \rangle = \langle LT(I) \rangle$.

Equivalently, but more informally, a set $\{g_1, \ldots, g_t\} \subset I$ is a Groebner basis of I if and only if the leading term of any element of I is divisible by one of the $LT(g_i)$.

In [12], Buchberger provided an algorithm for constructing a Groebner basis for a given ideal. This algorithm can also be used to determine whether a polynomial belongs to a given ideal.

A *reduced Groebner basis* G is a Groebner basis where the leading coefficients of polynomials in G are all 1 and no monomial of an element of G lies in the ideal generated by the leading terms of other elements of $G : \forall g \in G$ and no monomial of g is in $\langle LT(G - \{g\}) \rangle$.

The important result is that, for a fixed monomial ordering, any nonzero ideal has a unique reduced Groebner basis. The algorithm for finding a Groebner basis can easily be extended to output its reduced Groebner basis. Thus we will have a canonical symbolic representation for any ideal.

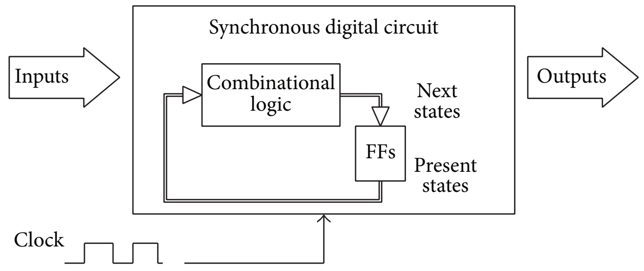


Figure 1: Synchronous digital circuits model.

Theorem 5 (the elimination theorem). *Let $I \subset k[x_1, \ldots, x_n]$ be an ideal and let G be a Groebner basis of I with respect to lexicographic order where $x_1 \succ x_2 \succ \cdots \succ x_n$. Then, for every $0 \le l \le n$, the set*

$$G_l = G \cap k\left[x_{l+1}, \ldots, x_n\right] \quad (1)$$

is a Groebner basis of the lth elimination ideal I_l.

Theorem 6. *Let G be a Groebner basis for an ideal $I \subset k[x_1, \ldots, x_n]$ and let $f \in k[x_1, \ldots, x_n]$. Then $f \in I$ if and only if the remainder on division of f by G is zero, denoted by $remd(f, G) = 0$.*

The property given in Theorem 6 can also be taken as the definition of a Groebner basis. Then we will get an efficient algorithm for solving the ideal membership problem. Assuming that we know a Groebner basis G for the ideal in question, we only need to compute the remainder with respect to G to determine whether $f \in I$.

In this paper, we will then use the ideal or any basis for the ideal as an efficient way of algebraization of system to be verified.

3. System Modeling with Polynomial

3.1. Circuit Representation Model. In this section, we will sketch the underlying digital system model for simulation used in our work.

Most modern circuit design is carried out within the synchronous model, which simplifies reasoning about behavior at the cost of strict requirements on clocking.

As shown in Figure 1, a classical synchronous design is comprised of combinatorial logic and blocks of registers with a global clock. In a synchronous digital system, the clock signal is often regarded as simple control signal and used to define a time reference for the movement of data within that system. Combinational logic performs all the logical functions in the circuit and it typically consists of logic gates. Registers usually synchronize the circuit's operation to the edges of the clock signal, and are the only elements which have memory properties.

The basic circuit model we used can be abstracted as the following model.

Definition 7 (synchronous circuit model). A synchronous circuit model structure is a tuple:

$$\mathbf{C} = \{\mathbf{clk}, \mathbf{L}, \mathbf{A}, \mathbf{Mux}, \mathbf{FFs}, \mathbf{I}, \mathbf{O}\}, \text{ where}$$

(i) **clk** is a global synchronous clock signal,
(ii) **L** is a set of logical operation units,
(iii) **A** is a set of arithmetic operation units,
(iv) **Mux** is a set of multiplex units,
(v) **FFs** is a set of sequential units,
(vi) **I** is a set of primary input signals,
(vii) **O** is a set of primary output signals.

In the following, we will discuss polynomial representation of a given circuit model. Firstly, let us retrospect the classical circuit representation model.

Traditionally, Binary decision diagrams (BDD) [13], the first generation decision diagrams technique, designed by Akers in 1978, acts as an efficient digital circuit representation (or Boolean functions). Recently, the next generation of decision diagrams, that is, word-level decision diagrams (WLDD) [14], has considerably widened its expressiveness for datapath operations due to processing of data in word-level format.

Generally, ROBDD or WLDD is mapped into hardware description languages according to the following scheme:

$$\textit{Function (circuit)} \Longleftrightarrow \textit{Model in ROBDD} \mid \textit{WLDD form.} \quad (2)$$

Though the dimension of a decision diagram is exponentially bounded by the number of variables, decision diagrams based canonical data structure is very useful in many well-known verification methods.

From the viewpoint of abstract symbolic computation, these decision diagrams based on representation for circuit systems are not suitable any more. Thus, we will adopt an alternative representation form based on polynomial sets instead of decision diagrams according to the following scheme:

$$\textit{Function (circuit)} \Longleftrightarrow \textit{Zero Set} \left(\textit{Polynomial Set}\right). \quad (3)$$

As is well known, given a monomial order, there is precisely one polynomial representation of a function.

For convenience, we introduce the following symbols for algebraic representations.

(1) For any symbolic (circuit, unit, signal, sequence, property, etc.) f, its algebraic representation form is denoted by $[\![f]\!]$.
(2) If a running cycle t is given, its algebraic representation form can be denoted by $[\![f]\!]_{[t]}$.
(3) Furthermore, if a time range $[m \cdots n]$ is specified, its algebraic representation form can then be denoted by $[\![f]\!]_{[t]}^{k}$ or $[\![f]\!]_{[t]}^{[m \cdots n]}$.

Here, t denotes the current time and $k = (n - m)$ denotes time steps.

Table 1: Polynomial model for arithmetic operation.

Arithmetic operation	Polynomial representation
$y = a + b$	$[\![+]\!] = (y - a - b)$
$y = a - b$	$[\![-]\!] = (y - a + b)$
$y = a * b$	$[\![*]\!] = (y - a * b)$
$y = a/b$	$[\![/]\!] = (y * b - a)$

Table 2: Polynomial model for logic operation.

Arithmetic operation	Polynomial representation
$y = \text{NOT } x$	$[\![\text{NOT}]\!] = (1 - x - y)$
$y = x_1 \text{ AND } x_2$	$[\![\text{AND}]\!] = (x_1 * x_2 - y)$
$y = x_1 \text{ OR } x_2$	$[\![\text{OR}]\!] = (x_1 + x_2 - x_1 * x_2 - y)$

Cycle-based symbolic simulation will be performed on the system model for verification. Intuitively, cycle-based symbolic simulation is a hybrid approach in the sense that the values that are propagated through the network can be either symbolic expressions or constant values. It assumes that there exists one unified clock signal in the circuit and all inputs of the systems remain unchanged while evaluating their values in each simulation cycle. The results of simulation report only the final values of the output signals or states in the current simulation cycle.

The detailed simulation process can be described as follows. Firstly, cycle-based symbolic simulation is initialized by setting the state of the circuit to the initial vector. Each of the primary input signals will be assigned a distinct symbolic or a constant value. Then, at the end of a simulation step, the expressions representing the next-state functions generally undergo a parametric transformation based optimization. After transformation, the newly generated functions are used as present state for the next state of simulation.

3.2. Arithmetic and Logic Unit Modeling. In this paper, we only focus on arithmetic unit for calculating fixed-point operations. For any arithmetic unit, integer arithmetic operations (addition, subtraction, multiplication, and division) can be constructed by the polynomials in Table 1.

The basic logic operations, like "**AND**", "**OR**", and "**NOT**" can be modeled by the following forms. Their corresponding polynomial representations [15] are specified as in Table 2.

Furthermore, we can extend the above rule to other common logic operators. For example,

$$y = x_1 \oplus x_2 \,(\text{or } y = x_1 \,\textbf{XOR}\, x_2)$$
$$\Longrightarrow [\![\oplus]\!] = (y - (x_1 + x_2 - x_1 * x_2) * (1 - x_1 * x_2)). \tag{4}$$

For all bit level variables x_i $(0 \le i \le n)$, a limitation $\langle\{x_i * x_i - x_i\}\rangle$ should be added.

3.3. Branch Unit Modeling. Basically, multiway branch is an important control structure in digital system. It provides a set of condition bits, bi $(0 \le i \le B)$, a set of target identifiers, $(0, \ldots, T-1)$, and a mapping from condition bit values to target identifiers. This mapping takes the form of a condition tree.

For any binary signal x, its value should be limited to $\{1, 0\}$ by adding $\{x * x - x\}$

$$y = Mux(x_0, x_1, \ldots, x_n, s),$$
$$i = s \longrightarrow y = x_i, \quad (0 \le i \le n)$$
$$\Longrightarrow [\![Mux]\!] = \left\langle \left\{ y - \sum_{i=1}^{n-1} \left(\prod_{j \in \{0,1,\ldots,n-1\} \setminus \{i\}} \left(\frac{(s-j)}{(i-j)} \right) \right) * x_i \right\} \right\rangle, \tag{5}$$

with $\prod_{i=0}^{n-1}(s-i) = 0$.

3.4. Sequential Unit Modeling. Each flip-flop (FF) in the circuit can be modeled as a multiplexer. We have the following proposition to state this model.

Proposition 8. *For a D flip-flop (D' is the next state), with an enable signal c, its equivalent combinational formal is* $y' = Mux(D, D', s) : i = s \rightarrow y' = x_i$, $(0 \le i < 2$, $x_0 = D$, $x_1 = D')$, *whose polynomial algebraic model can be described as* $[\![FFs]\!] =$

$$\langle (y' - D) * (c' - 1), (y' - D') * c, (y' - D) * (y' - D') \rangle \tag{6}$$

or

$$\langle y' - D * (c' - 1) - D' * c' \rangle. \tag{7}$$

Proof. Let D be the current state and let y' denote the next state of the flip-flop. When the signal c' value is 0, y' has the same value as D so that the FF maintains its present state; when the signal c' value is 1, y' takes a new value from the D' input (where, D' denotes the new value next state of the FF). Therefore, we have the 2-value multiway branch model and its polynomial set representation for FF. □

Proposition 9. *Let D be an FF model, (D' is the next state), without enable signal, then its equivalent combinational formal polynomial algebraic model can be described as:* $(y' - D)$.

Proof. Straightforward. □

3.5. Sequential Unrolling. Generally, for a sequential circuit **C**, one time frame of a sequential circuit is viewed as a combinational circuit in which each flip-flop will be converted into two corresponding signals: a pseudo primary input (PPI) and a pseudo primary output (PPO).

Symbolical simulation of a sequential circuit for n cycles can be regarded as unrolling the circuit n times. The unrolled circuit is still a pure combinational circuit, and the ith copy of the circuit represents the circuit at cycle i. Thus, the unrolled circuit contains all the symbolic results from the n cycles.

3.6. Indexed Polynomial Set Representation. To illustrate the sequential modeling for a given cycle number clearly, we define an *indexed polynomial set representation* for the ith cycle.

Let $xi_{[l]}$ $(0 \leq i \leq r)$ denote the input signals for the lth clock, $mi_{[l]}$ $(0 \leq i \leq s)$ the intermediate signals, and $yi_{[l]}$ $(0 \leq i \leq t)$ the output signals. We then have the following time frame expansion model for the sequential circuit:

$$FM = \left\{ \bigcup_{i=0}^{n} FM\,[i] \right\}, \tag{8}$$

where $FM[i] = \mathbf{C}(x1_{[i]}, \ldots, m1_{[i]}, \ldots, m1_{[i]}, \ldots, x1_{[i+1]}, \ldots, m1_{[i+1]}, \ldots, y1_{[i+1]}, \ldots)$ denote the ith time frame model.

Time frame expansion is achieved by connecting the PPIs (e.g., $x1_{[i+1]}$ from $FM[i+1]$) of the time frame to the corresponding PPOs ($x1_{[i+1]}$ from $FM[i]$) of the previous time frame.

4. Sequence Depth Calculation

In this subsection, we will discuss the important feature of SVA, time range, and its signal constraint unrolled model.

In SVA, for each sequence the earliest time step for the evaluation and the latest time step should be determined firstly. The sequence is then unrolled based on above information. Finally, the unrolled sequence will be performed using algebraization process.

In [3], a time range calculating algorithm is provided. Here, we will introduce some related definition and special handling for our purpose.

The following is the syntax definition for time range.

4.1. Time Range

Definition 10 (time range syntax). The syntax of "time range" can be described as follows.

cycle_delay_const_range_expression ::=

constant_expression : *constant_expression*

| *constant_expression* : \$.

Note that *constant_expression* is computed at compile time and must result in an integer value and can only be 0 or greater.

In this paper, we only focus on constant time range case. Thus, its form can be simplified as:

(1) $a\#\#[m:n]b\,(m, n \in \mathbf{N}$ and $n \geq m \geq 0)$

(2) $S_1\#\#[m:n]S_2\,(m, n \in \mathbf{N}$ and $n \geq m \geq 0)$.

Here, a, b are signals and S_1, S_2 are sequences.

Assume the starting time is cycle t, then we have: the sequence (1) will start $(n - m + 1)$ sequences of evaluation which are

$a\#\#mb$,

$a\#\#\,(m+1)b$,

$\ldots$,

$a\#\#(n-m+1)b$, respectively.

Their corresponding algebraic forms are

$[\![a\#\#mb]\!]_{[t]} = [\![a_t \wedge b_{t+1}]\!]$,

$[\![a\#\#(m+1)b]\!]_{[t]} = [\![a_t \wedge b_{t+2}]\!]$,

$\ldots$,

$[\![a\#\#(n-m+1)b]\!]_{[t]} = [\![a_t \wedge b_{t+n-m+1}]\!]$.

Then we have the equivalent form of above representation set as

$$[\![(a_t \wedge b_{t+m}) \vee \cdots \vee (a_t \wedge b_{t+n})]\!]. \tag{9}$$

4.2. Sequential Depth Calculation. The time range of a sequential is a time interval during which an operation or a terminal of a sequence has to be considered and is denoted by a closed bounded set of positive integers:

$$T = [l \cdots h] = \{x \mid l \leq x \leq h\} \quad (\text{here}, x, l, h \in \mathbf{N}). \tag{10}$$

Furthermore, the maximum of two intervals T_1 and T_2 is defined by $\max(T_1, T_2) = [\max(l_1, l_2) \cdots \max(h_1, h_2)]$.

In the same manner, the sum of two time ranges of T_1 and T_2 is defined as

$$T_1 + T_2 = [(l_1 + l_2) \cdots (h_1 + h_2)]. \tag{11}$$

Definition 11 (maximum sequential depth). The maximum sequential depth of a SVA expression F or a sequence, written as $dep(F)$, is defined recursively.

(i) $dep(a) = [1 \cdots 1]$, if a is a signal;

(ii) $dep(\neg a) = [1 \cdots 1]$, if a is a signal;

(iii) $dep(a\#\#b) = dep(a) + dep(b)$, if F_1, F_2 are sequences of SVA;

(iv) $dep(F_1\#\#[m:n]F_2) = dep(F_1) + dep(F_2) + [m \cdots n]$, if F_1, F_2 are sequences of SVA;

(v) $dep(F_1$ **and** $F_2) = \max(dep(F_1), dep(F_2))$, if F_1, F_2 are sequences of SVA;

(vi) $dep(F_1$ **or** $F_2) = \max(dep(F_1), dep(F_2))$, if F_1, F_2 are sequences of SVA;

(vii) $dep(F_1$ **intersect** $F_2) = dep(F_1) + dep(F_2) - 1$, if F_1, F_2 are sequences of SVA;

(viii) $dep(F[n]) = n * dep(F)$, if F is a sequence of SVA.

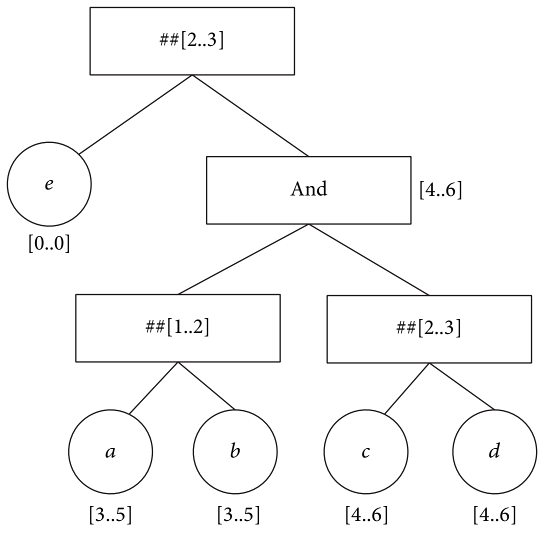


FIGURE 2: Sequence depth parsing tree.

For example, the following sequence is used to illustrate how to calculate the sequence depth.

sequence

$$e\ \#\#[2\cdots 3]\ ((a\#\#[1\cdots 2]b)\ \textbf{and}\ (c\ \#\#[1\cdots 2]\ d))$$

endsequence.

Figure 2 shows the binary parsing tree for calculating sequence depth of this sequence. The box denotes operator and cycle denotes signal node in the syntax tree.

Firstly, consider the top layer of the parsing tree and the sequence $\#\#[2\cdots 3]$. Here, because the first operand e is a terminal of the sequence, thus this operand becomes relevant at time step 0 only ($T_{1\text{st}} = [0\cdots 0]$). Since the interval has to be considered, the second operand (the subtree of "and") matches 2 or 3 time steps later. Therefore, calculating is recursively performed with $T = [0\cdots 0] + [2\cdots 3] = [2\cdots 3]$ for the second operand.

Similarly, we can have the sequence depth of the whole sequence and all its subsequences as denoted in the figure.

5. SVA to Polynomial Set Translation

As mentioned previously, SystemVerilog assertions are an integral component of SystemVerilog and provide two kinds of assertions: immediate assertions and concurrent assertions.

In this section, we only discuss concurrent assertions and their temporal layer representation model.

Concurrent assertions express functional design intent and can be used to express assumed input behavior, expected output behavior, and forbidden behavior. That is, assertions define properties that the design must meet. Many properties can be expressed strictly from variables available in the design, while properties are often constructed out of sequential behaviors.

Thus, we will firstly discuss the basic algebraization process for the sequential behavior model.

5.1. Algebraization Process. The properties written in SVA will be unrolled and checked against the design for bounded time steps in our method. Note that only a constrained subset of SVA can be supported by our method (unspecified upper bound time range and first-match operator are excluded).

Firstly, we translate the properties described by the constrained subset of SVA into flat sequences according to the semantics of each supported operator.

As mentioned in [16], the total set of SVA is divided into 4 subgroups, namely, simple sequence expression (SSE), interval sequence expression (ISE), complex sequence expression (CSE), and unbounded sequence expression (USE). Here, in our method, these groups can only be partly supported.

Therefore, we define the following sequence expressions by adding further conditions.

(i) Constrained simple sequence expression (CSSE) is formed by Boolean expression, **repeat** operator, and **cycle delay** operator.

(ii) Constrained interval sequence expression (CISE) is a super set of CSSE formed by extra time range operators.

(iii) Constrained complex sequence expression is a super set of CSSE and CISE containing operators **or** and **intersection**.

Secondly, the unrolled flat sequences will be added to temporal constraints to form proportional formulas with logical connectives ($\vee$, $\wedge$, and $\neg$).

Finally, the resulted proportional formulas will be translated into equivalent polynomial set.

Then, the verification problem is reduced to proving zero set inclusion relationship which can be resolved by Groebner bases approaches.

5.2. Boolean Layer Modeling. The Boolean layer of SVA forms an underlying basis for the whole assertion architecture which consists of Boolean expressions that hold or do not hold at a given cycle.

In this paper, we distinguish between signal logic values and truth logic values. That is, for a truth logic statement about a given property, its truth can be evaluated to *true* or *false*. But for a signal, when primary inputs are symbolic values, its signal logic value may not be evaluated as *high* or *low*.

Therefore, we have the following definition for signal logic.

Definition 12 (signal logic). In digital circuit systems, signal logic (**SL**, for short) is defined as:

(i) if a signal x is active-high (H, for short), then its signal value is defined as 1;

(ii) if a signal x is active-low (L, for short), then its signal value is defined as 0.

Definition 13 (symbolic constant). *A symbolic constant* is a rigid Boolean variable that forever holds the same Boolean value. The notion of symbolic constant was firstly introduced in STE [17] for two purposes:

(1) to encode an arbitrary Boolean constraints among a set of circuit nodes in a parametric form;

(2) to encode all possible scalar values for a set of nodes.

Assume H denotes a symbolic constant for signal logic and $\overline{H}$ denotes its negative form, if H denotes *high* then $\overline{H}$ will be *low*.

Consider ($req{==}\overline{H}$) and ($ack{==}H$) as an example. According to our definitions, *req* and *ack* are signals belonging to signal logic, while both ($req{==}\overline{H}$) and ($ack{==}H$) are of truth logic.

For example, assertion ($a[15 : 0] == b[15 : 0]$) is also a valid Boolean expression stating that the 16-bit vectors $a[15 : 0]$ and $b[15 : 0]$ are equal.

In SVA, the following are valid Boolean expressions:

(i) $arrayA == arrayB$

(ii) $arrayA! = arrayB$

(iii) $arrayA[i] >= array\ B[j]$

(iv) $arrayB[i][j+ : 2] == arrayA[k][m- : 2]$

(v) $(arrayA[i]\&(arrayB[j])) == 0$.

Since the state of a signal variable can be viewed as a zero of a set of polynomials. We have the following.

(1) For any signal x holds at a given time step i, thus, the state of $x == 1$ (x is active-high at cycle i) can be represented by polynomial $\{x_{[i]} - 1\}$.

(2) Alternatively, the state of $x == 0$ (x is active-low at cycle i) can be represented by polynomial $\{x_{[i]}\}$.

(3) Symbolically, the state of $x == H$ (x is active-high H at the ith cycle) can be modeled as $\{x_{[i]} - H\}$.

5.3. Sequence Operator Modeling. Temporal assertions define not only the values of signals, but also the relationship between signals over time. The sequences are the building blocks of temporal assertions and can express a set of linear behavior lasting for one or more cycles. These sequences are usually used to specify and verify interface and bus protocols.

A sequence is a regular expression over the Boolean expressions that concisely specifies a set of linear sequences. The Boolean expressions must be true at those specific clock ticks for the sequence to be true over time.

SystemVerilog provides several sequence composition operators to combine individual sequences in a variety of ways that enhance code writing and readability which can construct sequence expressions from Boolean expressions.

In this paper, **throughout** operator, [1 : $] operator, and the **first match** operator are not supported by our method.

SystemVerilog defines a number of operations that can be performed on sequences. The sequence composition operators in SVA are listed as follows.

Definition 14 (sequence operator).

R ::= b//"Boolean expression" form

| $(1, v = e)$// "local variable sampling" form

| (R)// "parenthesis" form

| (R_1 ##1 R_2)// "concatenation" form

| (R_1 ##0 R_2)// "fusion" form

| (R_1 **or** R_2)// "or" form

| (R_1 **intersect** R_2)// "intersect" form

| **first_match** (R)// "first match" form

| $R[*0]$// "null repetition" form

| $R[*1 : \$]$// "unbounded repetition" form

| $[*], [=], [->]$// "repetition" repeater

| **throughout**//specifying a Boolean expression must hold throughout a sequence

| **within**//specifying conditions within a sequence.

The resulted sequences constructed by operators are then used in properties for use in assertions and covers.

5.3.1. Cycle Delay Operator. In SystemVerilog, the ## construct is referred to as a cycle delay operator.

"##1" and "##0" are concatenation operators: the former is the classical regular expression concatenation; the latter is a variant with one-letter overlapping.

A ##n followed by a number n or range specifies the n cycles delay from the current clock cycle to the beginning of the sequence that follows.

(1) Fixed-length Case

sequence *fixs;*

a##nb

endsequence

$\Rightarrow [\![fixs]\!] = \{[\![a]\!]_t, [\![b]\!]_{t+n}\}$

(2) Time-range Case

sequence *tms*

$a\#\#[m \cdots n]b$

endsequence

$\Rightarrow [\![tms]\!] = \{[\![a]\!]_{[t]}, [\![b]\!]^m_{[t]}\} \vee \cdots \vee \{[\![a]\!]_{[t]}, [\![b]\!]^n_{[t]}\}$.

5.3.2. Intersect Operator. The two operands of intersect operator are sequences. The requirements for match of the intersect operation are as follows.

(i) Both operands must match.

(ii) The lengths of the two matches of the operand sequences must be the same.

R_1 **intersect** R_2.

R_1 starts at the same time as R_2; the intersection will match if R_1, starting at the same time as R_2, matches at the same time as R_2 matches.

Therefore, we have

$$[\![R_1 \textbf{ intersect } R_2]\!] = [\![R_1]\!]_{[t]}^{dep(R_1)} \wedge [\![R_2]\!]_{[t]}^{dep(R_2)}. \tag{12}$$

The sequence length matching intersect operator constructs a sequence like the and nonlength matching operator, except that both sequences must be completed in same cycle.

5.3.3. ***and*** *Operator.* R_1 **and** R_2.

This operator states that R_1 starts at the same time as R_2 and the sequence expression matches with the later of R_1 and R_2 matching. This binary operator and is used when both operands are expected to match, but the end times of the operand sequences can be different.

That is, R_1 **and** R_2 denotes both R_1 and R_2 holds for the same number cycles. Then, the matches of R_1 **and** R_2 must satisfy the following:

(i) The start point of the match of R_1 must be no earlier than the start point of the match of R_2.

(ii) The end point of the match of R_1 must be no later than the end point of the match of R_2.

The sequence nonlength matching **and** operator constructs a sequence in which two sequences both hold at the current cycle regardless of whether they are completed in the same cycle or in different cycles

$$[\![R_1 \textbf{ and } R_2]\!] \Longrightarrow \bigvee_{i=l_1}^{h_1}\bigvee_{j=l_2}^{h_2} \left(\left\{[\![R_1]\!]_{[t]}^{i}\right\} \wedge \left\{[\![R_2]\!]_{[t]}^{j}\right\}\right). \tag{13}$$

Here, $l_i = dep(R_i) \cdot l$, $h_i = dep(R_i) \cdot h$, and $0 < i \leq 2$.

5.3.4. ***or*** *Operator.* R_1 **or** R_2.

The sequence **or** operator constructs a sequence in which one of two alternative sequences hold at the current cycle. Thus, the sequence $(a\#\#1b)$ **or** $(c\#\#1d)$ states that either sequence a, b or sequence c, d would satisfy the assertion

$$[\![R_1 \textbf{ or } R_2]\!] \Longrightarrow \bigvee_{i=l_1}^{h_1}\bigvee_{j=l_2}^{h_2} \left(\left\{[\![R_1]\!]_{[t]}^{i}\right\} \vee \left\{[\![R_2]\!]_{[t]}^{j}\right\}\right). \tag{14}$$

Here, $l_i = dep(R_i) \cdot l$, $h_i = dep(R_i) \cdot h$, and $0 < i \leq 2$.

5.3.5. Local Variables. SystemVerilog provides a feature by which variables can be used in assertions. The user can declare variables local to a property. This feature is highly useful in pipelined designs where the consequent occurrence might be many cycles later than their corresponding antecedents.

Local variables are optional and local to properties. They can be initialized, assigned (and reassigned) a value, operated on, and compared to other expressions.

The syntax of a sequence declaration with a local variable is shown below.

> *sequence_declaration* ::=
> **sequence** *sequence_identifier*[([*tf_port_list*])];
>
> > *assertion_variable_declaration*
> > *sequence_expr*;
>
> **endsequence** [: *sequence_identifier*].

The property declaration syntax with a local variable can be illustrated as follows:

> *property_declaration* ::=
> **property** *property_identifier*[([*tf_port_list*])];
>
> > $assertion_variable\_declaration$
> > *property_spec*;
>
> **endproperty** [: *property_identifier*].

The variable identifier declaration syntax of a local variable can be illustrated as follows:

> *assertion_variable_declaration* ::=
> *var_data_type list_of_variable_identifiers*.

The dynamic creation of a variable and its assignment is achieved by using the local variable declaration in a sequence or property declaration and making an assignment in the sequence.

Thus, local variables of a sequence (or property) may be set to a value, which can be computed from a parameter or other objects (e.g., arguments, constants, and objects visible by the sequence (or property)).

For example, a property of a pipeline with a fixed latency can be specified below.

> **property** *latency*;
> *int x*;
> $(valid\_in, x = p\_in)|-> \#\#3(p\_out == (x + 1))$;
> **endproperty**.

This property e is evaluated as follows:

When *valid_in* is **true**, x is assigned the value of *p_in*. If 3 cycles later, *p_out* is equal to $x + 1$, then property *latency* is **true**. Otherwise, the property is **false**. When *valid_in* is **false**, property *latency* is evaluated as **true**.

For the algebraization of SVA properties with local variables, in our method these local variables will be taken as common signal variables (symbolic constant) without any sequential information.

Thus, we have the polynomial set representation for property *latency*:

$$
\begin{aligned}
&[\![latency]\!]_{[t]} = \{\\
&\qquad (valid\_in_{[t]} - 1),\\
&\qquad (x - p\_in_{[t]}),\\
&\qquad (p\_out_{[t+3]} - (x+1))\}.
\end{aligned}
$$

5.3.6. Repetition Operators. SystemVerilog allows the user to specify repetitions when defining sequences of Boolean expressions. The repetition counts can be specified as either a range of constants or a single constant expression.

Nonconsecutive repetition specifies finitely many iterative matches of the operand Boolean expression, with a delay of one or more clock ticks from one match of the operand to the next successive match and no match of the operand strictly in between. The overall repetition sequence matches at or after the last iterative match of the operand, but before any later match of the operand.

The syntax of repetition operator can be illustrated as follows:

$$non\_consecutive\_rep ::= [= const\_or\_range\_expr]. \quad (15)$$

The number of iterations of a repetition can be specified by exact count.

For example, a sequence with repetition can be defined as

$$a \mid => b\,[= 3]\ \#\#1\ c. \quad (16)$$

The sequence expects that 2 clock cycles after the valid start, signal "*b*" will be repeated three times.

We have $[\![R[= n]]\!] = \bigvee_{i=0}^{n}(\{[\![R]\!]_{[t]}^{i*dep(R)}\})$.

5.4. Property Operator Modeling. In general, property expressions in SVA are built using sequences, other sublevel properties, and simple Boolean expressions via property operators.

In SVA, a property that is a sequence is evaluated as true if, and only if, there is a nonempty match of the sequence. A sequence that admits an empty match is not allowed as a property.

The success of assertion-based verification methodology relies heavily on the quality of properties describing the intended behavior of the design. Since it is a fairly new methodology, verification and design engineers are often faced with a question of "how to identify good properties" for a design. It may be tempting to write properties that closely resemble the implementation.

Properties we discussed in this paper can be classified as.

(1) Design Centric. Design centric properties represent assertions added by the RTL designers to characterize white box design attributes. These properties are typically towards FSMs, local memories, clock synchronization logic, and other hardware-related designs. The design should conform to the expectations of these properties.

(2) Assumption Centric. Assumption centric properties represent assumptions about the design environment. They are used in informal verification to specify assumptions about the inputs. The assume directive can inform a verification tool to assume such a condition during the analysis.

(3) Requirement/Verification Centric. Many requirement properties are best described using a "cause-to-effect" style that specifies what should happen under a given condition. This type of properties can be specified by implication operator. For example, a handshake can be described as "if request is asserted, then acknowledge should be asserted within 5 cycles".

In general, property expressions are built using sequences, other sublevel properties, and simple Boolean expressions.

These individual elements are combined using property operators: **implication**, **NOT**, **AND**, **OR**, and so forth.

The property composition operators are listed as follows.

Definition 15 (property operator).

$$
\begin{aligned}
P ::= {} & R /^{*}\ \text{"sequence" form}\ ^{*}/ \\
& |(P) /^{*}\ \text{"parenthesis" form}\ ^{*}/ \\
& |\mathbf{not}\ P /^{*}\ \text{"negation" form}\ ^{*}/ \\
& |(P_1\ \mathbf{or}\ P_2) /^{*}\ \text{"or" form}\ ^{*}/ \\
& |(P_1\ \mathbf{and}\ P_2) /^{*}\ \text{"and" form}\ ^{*}/ \\
& |(R|->P)\,|(R|=>P) /^{*}\ \text{"implication" form}\ ^{*}/ \\
& |\mathbf{disable\ iff}(b)\ P /^{*}\ \text{"reset" form}\ ^{*}/.
\end{aligned}
$$

Note that **disable if and only if** will not be supported in this paper. Property operators construct properties out of sequence expressions.

5.4.1. Implication

Operators. The SystemVerilog implication operator supports sequence implication and provides two forms of implication: overlapped using operator |− >, and nonoverlapped using operator | =>, respectively.

The syntax of implication is described as follows:

$$
\begin{aligned}
& sequence\_expr \mid - > property\_expr \\
& sequence\_expr \mid => property\_expr.
\end{aligned}
\quad (17)
$$

The implication operator takes a sequence as its antecedent and a property as its consequent. Every time the sequence matches the property must hold. Note that both the running of the sequence and the property may span multiple clock cycles and that the sequence may match multiple times.

For each successful match of the antecedent sequence, the consequence sequence (right-hand operand) is separately evaluated, beginning at the end point of the matched antecedent sequence. All matches of antecedent sequence require a match of the consequence sequence.

Moreover, if the antecedent sequence (left hand operand) does not succeed, implication succeeds vacuously by returning **true**. For many protocol related assertions, it is important to specify the sequence of events (the antecedent or cause) that must occur before checking for another sequence (the consequent or effect). This is because if the antecedent does not occur, then there is no need to perform any further verification. In hardware, this occurs because the antecedent reflects an expression that, when active, triggers something to happen.

A property with a consequent sequence behaves as if the consequent has an implication first-match applied to it.

*5.4.2. **NOT***

Operator. The operator **NOT** *property_expr* states that the evaluation of the property returns the opposite of the evaluation of the underlying *property_expr*.

*5.4.3. **AND***

Operator. The formula "P_1 **AND** P_2" states that the property is evaluated as **true** if, and only if, both P_1 and P_2 are evaluatd as **true**.

*5.4.4. **OR***

Operator. The operator "P_1 **OR** P_2" states that the property is evaluated as **true** if and only if, at least one of P_1 and P_2 is evaluated as **true**.

*5.4.5. **IF-ELSE***

Operator. This operator has two valid forms which are listed as follows.

(1) **IF** $(dist)$ P_1.

A property of this form is evaluated as **true** if, and only if, either *dist* is evaluated as **false** or P_1 is evaluated as **true**.

(2) **IF** $(dist)$ P_1 **ELSE** P_2.

A property of this form is evaluated as **true** if, and only if, either *dist* is evaluated as **true** and P_1 is evaluated as **true** or *dist* is evaluated as **false** and P_2 is evaluated as **true**.

From previous discussion, we have the following proposition for property reasoning.

Proposition 16. *Assume that P, P_1, and P_2 are valid properties in SVA, the following rules are used to construct the corresponding verification process. Here, $AssChk(Model, Property)$: $\{\textbf{ture}, \textbf{false}\}$ is a self-defined checking function that can determine whether a given "Property" holds or not with respect to a circuit model "Model".*

(1) P_1 **OR** $P_2 \Rightarrow AssChk(M, [\![P_1]\!]) \vee AssChk(M, [\![P_2]\!])$

(2) P_1 **AND** $P_2 \Rightarrow AssChk(M, [\![P_1]\!]) \wedge AssChk(M, [\![P_2]\!])$

(3) **NOT** $P \Rightarrow \neg AssChk(M, [\![P]\!])$

(4) **IF** $(dist)$ $P_1 \Rightarrow \neg AssChk(M, dist) \vee AssChk(M, [\![P_1]\!])$

(5) **IF** $(dist)$ P_1 **ELSE** $P_2 \Rightarrow (AssChk(M, dist) \wedge AssChk(M, [\![P_1]\!])) \vee (\neg AssChk(M, dist) \wedge AssChk(M, [\![P_2]\!]))$

(6) $S \mid - > P \Rightarrow if\,(\neg AssChk(M, [\![S]\!]))return$ **true** $else$ $(AssChk(M, [\![S]\!]_{[t]}) \wedge AssChk(M, [\![P]\!]^{dep(P)}_{[t+dep(S)-1]}))$

(7) $S \;\; |=> \; P \Rightarrow if(\neg AssChk(M, [\![S]\!]))return$ **true** $else$ $(AssChk(M, [\![S]\!]_{[t]}) \wedge AssChk(M, [\![P]\!]^{dep(P)}_{[t+dep(S)]}))$.

6. Verification Algorithm

In this section, we will describe how an assertion is checked using Groebner bases approach. Firstly, we will discuss a practical algorithm using Groebner bases for assertion checking.

6.1. Basic Principle. As just mentioned, our checking method is based on algebraic geometry which is the study of the geometric objects arising as the common zeros of collections of polynomials. Our aim is to find polynomials whose zeros correspond to pairs of states in which the appropriate assignments are made.

We can regard any set of points in k^n as the variety of some ideal. We can then use the ideal or any basis for the ideal as a way of encoding the set of points.

From Groebner Bases theory [10, 12] every nonzero ideal $I \subset k[x_1, \ldots, x_n]$ has a Groebner basis and the following proposition evidently holds.

Proposition 17. *Let C and S be polynomial sets of $k[x_1, \ldots, x_n]$, and $\langle GS\rangle$ a Groebner basis for $\langle S\rangle$, and then we have that*

$$\langle C\rangle \subseteq \langle S\rangle \Leftrightarrow \forall c \in C : remd(c, GS) == 0 \text{ holds.}$$

Theorem 18. *Suppose that $A([\![A]\!] = \{a_1, a_2, \ldots, a_r\})$ is a property to be verified. T is the initial preconditions of C, and M is a system model to be checked. Let $[\![T]\!]$ and $[\![M]\!]$ be the polynomial set representations for T and M, respectively, constructed by previous mentioned rules. Let $H = [\![T \cup M]\!] = \{h_1, h_2, \ldots, h_s\} \subseteq k[x_1, \ldots, x_n]$, $I = \langle H\rangle$ (where, $\langle H\rangle$ denotes the ideal generated by H) and $GB_H = gbasis(H, \prec)$, then we have*

$$\begin{aligned} &\left((1 \notin GB_H) \textbf{ and } remd\,(C, GB_H) == 0\right) \\ &\quad \Longleftrightarrow \left((1 \notin GB_H) \textbf{ and } \bigwedge_{i=0}^{r} (remd\,(g_i, GB_H) == 0)\right) \\ &\quad \Longleftrightarrow (M \models A) \text{ holds.} \end{aligned} \quad (18)$$

Proof. (1) The polynomial set $[\![M]\!]$ of system model M describes the data-flow relationship that inputs, outputs, and functional transformation should meet. The initial condition $[\![M]\!]$ can be seen as extra constraint applied by users. There should not exist contradiction between them; that is, the polynomial equations of $[\![T \cup M]\!]$ must have a common solution.

By Hilbert's Nullstellensatz theory, if we have polynomials $[\![T \cup M]\!] = \{f_1, \ldots, f_s\} \in k[x_1, \ldots, x_n]$, we compute a reduced Groebner basis of the ideal they generate with respect to any ordering. If this basis is $\{1\}$, the polynomials have no common zero in k^n; if the basis is not $\{1\}$, they must have a common zero.

Therefore, $(1 \notin GB_H)$ should hold.

(2) Evidently, by previous proposition, it is easy to have that $remd(C, GB_H) == 0$ should hold.

Thus, we have that this theorem holds. $\square$

```
Input:
(1) circuit model $M$;
(2) initial condition $T$;
(3) an assertion $A$;
Output: Boolean: true or false;
BEGIN
    /* Step 1: initialize input signals via testbench */
(00)    $InitSignals(\vec{T})$;
(01)    $\mathscr{M} = \emptyset$; $PS_A = \emptyset$; $H = \emptyset$; $PS_T = \emptyset$;
    /* Step 2: build polynomial model */
(02)    $\mathscr{M} = [\![M]\!] = Build\mathrm{PS}(M)$;
    /* Step 3: build polynomial set for initial condition $T$ */
(03)    $PS_T = [\![T]\!] = Build\mathrm{PS}(T)$;
    /* Step 4: build polynomial set for consequent $\mathscr{A}$ */
(04)    $PS_A = [\![A]\!] = Build\mathrm{PS}(A)$;
    /* Step 5: calculate the $PS_T \cup \mathscr{M}$ */
(05)    $H = PS_{\mathscr{T}} \cup \mathscr{M}$;
    /* Step 6: calculate the Groebner base of $\langle H \rangle$ */
(06)    $GB_H := gbasis(H, \prec)$;
    /* Step 7: determine the basis is {1} or not */
(07) if($1 \in GB_H$) {
(08)            return false; }
    /* Step 8: check every polynomial */
    /* $A = \{a_1, a_2, \dots, a_{|A|}\}$ */
(09) $i = 1$;
(10) while($i \le |A|$){
(11)     if($remd([\![a_i]\!], GB_H) \neq 0$) {
(12)            return false; }
(13)         $i$++;
(14)     } /* endwhile */
(15)    return true; /* Assertion does hold */
END;
```

Algorithm 1: "*AssChkPolyBuild*(M, T, A)".

6.2. Checking Algorithm. For simplicity, in this section, we only provide the key decision algorithm.

Firstly, the original system model is transformed into a normal polynomial representation and the assertion as well. Then, calculate the hypothesis set and its Groebner basis using the Buchberger algorithm [18] and their elimination ideals. Finally, examine the inclusion relationship between elimination ideals to determine whether the assertion to be checked holds or not.

From above discussion, we have the following process steps and detailed algorithm description. Algorithm 1.

For convenience, we can derive another version checking procedure named "*AssChk*$([\![M]\!], [\![A]\!])$" which can accept polynomial representations as inputs without explicit polynomial construction process.

Further, by applying Theorem 18 and the checking algorithm "*AssChk*$([\![M]\!], [\![A]\!])$", we can easily verify all supported properties written in SVA.

7. An Example

In this section, we will study a classical circuit to show how SVA properties are verified by polynomial representation and algebra computation method.

Consider the Johnson counter circuit in Figure 3. Johnson counters can provide individual digit outputs rather than a binary or BCD output, as shown in Table 3. Notice that each legal count may be defined by the location of the last flip-flop to change states and which way it changed state.

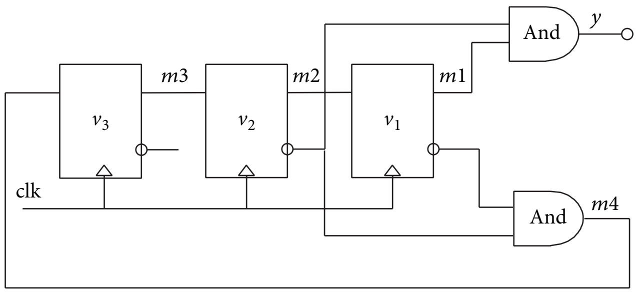


Figure 3: Johnson counter.

Table 3: Table of Johnson counter output.

Counter (clock)	$v3(m3)$	$v2(m2)$	$v1(m1)$
0	0	0	0
1	1	0	0
2	1	1	0
3	0	1	1
4	0	0	1
5	0	0	0

7.1. Algebraization of Circuit and Assertion. Johnson counter provides individual digit outputs, as shown in Table 3.

The polynomial set for Johnson counter circuit in Figure 3 can be constructed as follows:

$$\begin{aligned} Set_{\text{adder}} = \{ & f1 = \{m3' - m4\}, f2 = \{m2' - m3\}, \\ & f3 = \{m1' - m2\}, \\ & f4 = \{m4 - (1 - m1) * (1 - m2)\}, \\ & f5 = \{y - m1 * (1 - m2)\}\}, \end{aligned} \tag{19}$$

where $m1'$ denotes the next state of $m1$. For the ith cycle, we use $x1_{[i]}$ to denote variable name in current cycle.

Similarly, another Johnson counter circuit with an error is shown in Figure 4, the corresponding polynomial set can be described by

$$\begin{aligned} Set_{\text{adder}} = \{ & f1, f2, f3, \\ & f4' = \{m4 + m1 + m2 \\ & \quad - (1 - m1) * (1 - m2)\}, f5\}. \end{aligned} \tag{20}$$

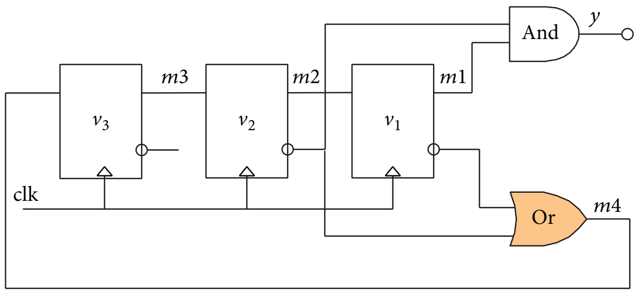


Figure 4: Johnson counter with error.

To illustrate the problem clearly, we define polynomial set representation $PM[i]$ for ith cycle as follows:

$$\begin{aligned} PM[i] = \{ & m3_{[i+1]} - m4_{[i]}, m2_{[i+1]} - m3_{[i]}, \\ & m1_{[i+1]} - m2_{[i]}, m4_{[i]} \\ & - (1 - m2_{[i]}) * (1 - m1_{[i]}), \\ & y - m1_{[i]} * (1 - m2_{[i]}) \}. \end{aligned} \tag{21}$$

Support that the circuit will run 6 cycles for verification, therefore, we have $PM = \{\bigcup_{i=0}^{5} PM[i]\}$.

For any Boolean variable a, we will add an extra constraint: $a*a-a$. Thus, we define the corresponding constraints set as follows: $CNS[i] = \{c_{[i]} * c_{[i]} - c_{[i]}, c \in \{m1, m2, m3, m4, y\}\}$.

In the same manner, we have $CNS = \{\bigcup_{i=0}^{5} CNS[i]\}$.

The sequential property for this counter circuit can be specified by the following SystemVerilog assertions.

property $JCPA$;

$(m3 = 0 \ \&\& \ m2 = 0 \ \&\& \ m1 = 0)$
$| => \#\#1(m3 == 1 \ \&\& \ m2 == 0 \ \&\& \ m1 == 0)$
$| => \#\#2(m3 == 1 \ \&\& \ m2 == 1 \ \&\& \ m1 == 0)$
$| => \#\#3(m3 == 0 \ \&\& \ m2 == 1 \ \&\& \ m1 == 1)$;

endproperty

property $JCP1$;

$@(clk)(m1 == 0)| => \#\#2 \ (m1 == 0)$;

endproperty

property $JCP2$;

$@(clk)(m2 == 0)| => \#\#2 \ (m2 == 1)$;

endproperty

property $JCP3$;

$@(clk)(m3 == 0)| => \#\#2 \ (m3 == 1)$;

endproperty.

Table 4: Polynomial forms.

Name	Precondition	Expected consequent
JCPA	$\{m1_{[0]}, m2_{[0]}, m3_{[0]}\}$	
	cycle 1	$\{m1_{[1]} - 1, m2_{[1]}, m3_{[1]}\}$
	cycle 2	$\{m1_{[2]} - 1, m2_{[2]} - 1, m3_{[2]}\}$
	cycle 3	$\{m1_{[3]}, m2_{[3]} - 1, m3_{[3]} - 1\}$
JCP1	$\{m1_{[0]}\}$	$m1_{[2]}$
JCP2	$\{m2_{[0]}\}$	$m2_{[2]} - 1$
JCP3	$\{m3_{[0]}\}$	$m3_{[2]} - 1$

We will afterwards demonstrate the verification process step by step.

The circuit model to be verified is as shown below:

$$SM = PM \cup CNS. \tag{22}$$

The property of this counter can be specified as the following SVA assertion:

assert property $(JCP1)$
assert property $(JCP2)$.

The Algebraization form of the above properties can be modeled as in Table 4.

7.2. Experiment Using Maple. We run this example by using Maple 13. Before running, we manually translated all models into polynomials. The experiment is performed on a PC with a 2.40 GHz CPU (intel i5 M450) and 1024 MB of memory. It took about 0.04 seconds and 0.81 MB of memory when applying Groebner method.

[> $with(Groebner)$
[> $CM := \cdots /{}^{*}$Circuit Model$^{*}/$
[> $TDEG := tdeg($

$m1_{[0]}, m2_{[0]}, m3_{[0]}, m4_{[0]},$
$m1_{[1]}, m2_{[1]}, m3_{[1]}, m4_{[1]},$
$m1_{[2]}, m2_{[2]}, m3_{[2]}, m4_{[2]},$
$m1_{[3]}, m2_{[3]}, m3_{[3]}, m4_{[3]},$
$m1_{[4]}, m2_{[4]}, m3_{[4]}, m4_{[4]},$
$y_{[0]}, y_{[1]}, y_{[2]}, y_{[3]})$

[> $CGB := Basis(CM, TDEG)$.

Thus, we can have the Groebner basis as follows:

[> $[y1_{[3]}, y1_{[2]}, y1_{[1]}, y1_{[1]}, m3_{[4]}, m2_{[4]},$

$m1_{[4]} - 1, m4_{[3]}, m3_{[3]}, m2_{[3]} - 1, m1_{[3]} - 1,$
$m4_{[2]}, m3_{[2]} - 1, m2_{[2]} - 1, m1_{[2]}, m4_{[1]} - 1,$
$m3_{[1]} - 1,$
$m2_{[1]}, m1_{[1]}, m4_{[0]} - 1, m3_{[0]}, m2_{[0]}, m1_{[0]}]$

[> $ret := NormalForm(m3_{[1]} - 1, CGB, TDEG)$
[> $ret = 0$.

Table 5: Computation result table.

Number	Polynomial	GBase	Mem	Time	Return	Result
	$\{m1_{[1]}\}$	$1 \notin CGB$			0	
0	$\{m2_{[1]}\}$	$1 \notin CGB$	0.80 M	0.02 S	0	Yes
	$\{m3_{[1]} - 1\}$	$1 \notin CGB$			0	
	$\{m1_{[2]}\}$	$1 \notin CGB$			0	
1	$\{m2_{[2]} - 1\}$	$1 \notin CGB$	0.81 M	0.03 S	0	Yes
	$\{m3_{[2]} - 1\}$	$1 \notin CGB$			0	
	$\{m1_{[3]} - 1\}$	$1 \notin CGB$			0	
2	$\{m2_{[3]} - 1\}$	$1 \notin CGB$	0.81 M	0.04 S	0	Yes
	$\{m3_{[3]}\}$	$1 \notin CGB$			0	
3	$\{m1_{[2]}\}$	$1 \notin CGB$	0.81 M	0.04 S	0	Yes
4	$\{m2_{[2]}\}$	$1 \notin CGB$	0.81 M	0.04 S	0	Yes
5	$\{m3_{[2]}\}$	$1 \notin CGB$	0.81 M	0.04 S	0	Yes

Table 6: Computation result table.

Number	Polynomial	GBase	Mem	Time	Return	Result
	$\{m1_{[1]}\}$	$1 \notin CGB$			0	
0	$\{m2_{[1]}\}$	$1 \notin CGB$	0.80 M	0.02 S	0	Yes
	$\{m3_{[1]} - 1\}$	$1 \notin CGB$			0	
	$\{m1_{[2]}\}$	$1 \notin CGB$			0	
1	$\{m2_{[2]} - 1\}$	$1 \notin CGB$	0.81 M	0.03 S	0	Yes
	$\{m3_{[2]} - 1\}$	$1 \notin CGB$			0	
	$\{m1_{[3]} - 1\}$	$1 \notin CGB$			0	
2	$\{m2_{[3]} - 1\}$	$1 \notin CGB$	0.81 M	0.14 S	0	NO
	$\{m3_{[3]}\}$	$1 \notin CGB$			−1	

As shown in maple outputs, the given circuit has been modeled as polynomial set CM (its Groebner bases is denoted by CGB) and assertion expected result as $\{m3_{[1]} - 1\}$. From the running result, we have $1 \notin CGB$ and return value of *NormalForm* is 0 which means CGB is divided with no remainder by $\{m3_{[1]} - 1\}$.

Thus, from the previously mentioned verification principles, it is easy to conclude that the assertion $JCP1$ holds under this circuit model after 3 cycles.

More detailed experiment results for verification of circuit shown in Figure 3 are listed in Table 5.

Conversely, we check these assertions against the circuit shown in Figure 4 in a similar way. More detailed experiment results are demonstrated in Table 6.

From above table, when checking $JCPA$ assertion, the result $ret := NormalForm(m3_{[3]}, CGB, TDEG) \neq 0$ so that we can conclude the assertion does not hold. Therefore, there must exist unexpected errors in the original circuit.

This case is a fairly complete illustration of how the checking algorithm works.

8. Conclusion

In this paper, we presented a new method for SVA properties checking by using Groebner bases based symbolic algebraic approaches. To guarantee the feasibility we defined a constrained subset of SVAs, which is powerful enough for practical purposes.

We first introduce a notion of symbolic constant without any sequential information for handling local variables in SVAs inspired from STE. We then proposed a practical algebraization method for each sequence operator. For sequential circuits verification, we introduce a parameterized polynomial set modeling method based on time frame expansion.

Our approach is based on polynomial models construction for both circuit models and SVA assertions. This method is to eventually translate a simulation based verification problem into a pure algebraic zero set determination problem by a series of proposed steps, which can be performed on any general symbolic algebraic tool.

This method allows users to deal with more than one state and many input combinations every cycle. This advantage comes directly from the fact that many vectors are simulated at once using symbolic value.

In summary, in this research, by suitable restrictions of SVA assertions, we can guarantee the availability of polynomial set representation. Based on this polynomial set model, symbolic simulation can be performed to produce symbolic traces and temporal relationship constraints of signal variables as well. We then apply symbolic algebra approach to check the zeros set relation between their polynomial sets and determine whether the temporal assertion holds or not under current running cycle.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

The project is supported by the National Natural Science Foundation of China under Grant no. 11371003, the Natural Science Foundation of Guangxi under Grants nos. 2011GXNSFA018154, 2012GXNSFGA060003, and 2013GXNSFAA019342, the Science and Technology Foundation of Guangxi under Grant no. 10169-1, the Scientific Research Project no. 201012MS274 from Guangxi Education Department, the Fundamental Research Funds for the Central Universities under Grants nos. DUT14QY05 and HCIC201204 of Guangxi Key Laboratory of Hybrid Computation and IC Design Analysis Open Fund, and the Baguio scholarship Project of Guangxi. The authors would like to thank their colleagues for participating in the research. They also appreciate the anonymous reviewers for their helpful comments.

References

[1] IEEE System Verilog Working Group, IEEE Standard for SystemVerilog C Unified Hardware Design, Specification, and Verification (IEEE Std 1800-2005). IEEE, 2005.

[2] "IEEE draft standard for system verilog—unified hardware design, specification, and verification language," IEEE P1800/D3, 2011.

[3] R. Wille, G. Fey, M. Messing, G. Angst, L. Linhard, and R. Drechsler, "Identifying a subset of SystemVerilog assertions for efficient bounded model checking," in *Proceedings of the 11th EUROMICRO Conference on Digital System Design Architectures, Methods and Tools (DSD '08)*, pp. 542–549, September 2008.

[4] L. Darringer, "Application of program verification techniques to hardware verification," in *Proceedings of the IEEE-ACM Design Automation Conference*, pp. 375–381, 1979.

[5] G. S. Avrunin, "Symbolic model checking using algebraic geometry," in *Proceedings of the 8th International Conference on Computer Aided Verification*, vol. CAV96, pp. 26–37, Springer, London, UK, 1996.

[6] W. B. Mao and J. Z. Wu, "Application of Wus method to symbolic model checking," in *Proceedings of the 2005 international Symposium on Symbolic and Algebraic Computation (ISSAC '05)*, pp. 237–244, ACM Press, Beijing, China, 2005.

[7] J. Wu and L. Zhao, "Multi-valued model checking via groebner basis approach," in *Proceedings of the 1st Joint IEEE/IFIP Symposium on Theoretical Aspects of Software Engineering (TASE '07)*, pp. 35–44, IEEE Computer Society Press, June 2007.

[8] N. Zhou, J. Wu, and X. Gao, "Algebraic verification method for SEREs properties via Groebner bases approaches," *Journal of Applied Mathematics*, vol. 2013, Article ID 272781, 10 pages, 2013.

[9] X. Gao, N. Zhou, J. Wu, and D. Li, "Wu's characteristic set method for SystemVerilog assertions verification," *Journal of Applied Mathematics*, vol. 2013, Article ID 740194, 14 pages, 2013.

[10] J. Little, D. Cox, and D. O'Shea, *Ideals, Varieties, and Algorithms*, Springer, New York, NY, USA, 1992.

[11] T. Becker and V. Weispfenning, *Gröbner Bases: A Computational Approach to Commutative Algebra*, Springer, New York, NY, USA, 1993.

[12] B. Buchberger, "Groebner bases: an algorithmic method in polynomial ideal theory," in *Multidimensional Systems Theory*, pp. 184–232, 1985.

[13] S. B. Akers, "Binary decision diagrams," *IEEE Transactions on Computers C*, vol. 27, no. 6, pp. 509–516, 1978.

[14] S. Hoereth and R. Drechsler, "Formal verification of word-level specifications," in *Proceedings of the Design, Automation and Test in Europe Conference and Exhibition*, pp. 52–58, Munich, Germany, 1999.

[15] Y. M. Ryabukhin, "Boolean ring," in *Encyclopaedia of Mathematics*, Springer, Michiel, Germany, 2001.

[16] S. Das, R. Mohanty, P. Dasgupta, and P. P. Chakrabarti, "Synthesis of system verilog assertions," in *Proceedings of the Conference on Design, Automation and Test in Europe: Designers' Forum (DATE '06)*, pp. 70–75, European Design and Automation Association, Leuven, Belgium, March 2006.

[17] C.-J. Seger and R. E. Bryant, "Formal verification by symbolic evaluation of partially-ordered trajectories," *Formal Methods in System Design*, vol. 6, no. 2, pp. 147–190, 1995.

[18] D. Cox, J. Little, and D. O'Shea, *Ideals, Varieties, and Algorithms An Introduction to Computational Algebraic Geometry and Commutative Algebra*, Undergraduate Texts in Mathematics, Springer, New York, 3rd edition, 2007.

Hindawi Publishing Corporation
Journal of Applied Mathematics
Volume 2014, Article ID 901539, 11 pages
http://dx.doi.org/10.1155/2014/901539

Research Article

A Nonlinear Multiparameters Temperature Error Modeling and Compensation of POS Applied in Airborne Remote Sensing System

Jianli Li,[1,2] Wenjian Wang,[1,2] Feng Jiao,[1,2] Jiancheng Fang,[1,2] and Tao Yu[3]

[1] *School of Instrument Science and Opto-Electronic Engineering, Beijing University of Aeronautics and Astronautics, Beijing 100191, China*
[2] *Science & Technology on Inertial Laboratory, Key Laboratory of Fundamental Science for National Defense-Novel Inertial Instrument & Navigation System Technology, Beijing University of Aeronautics and Astronautics, Beijing 100191, China*
[3] *Xi'an Institute of Optics and Precision Mechanics (XIOPM), CAS, Xi'an 710119, China*

Correspondence should be addressed to Wenjian Wang; desa001@sina.com

Received 10 March 2014; Accepted 13 May 2014; Published 9 June 2014

Academic Editor: Guiming Luo

The position and orientation system (POS) is a key equipment for airborne remote sensing systems, which provides high-precision position, velocity, and attitude information for various imaging payloads. Temperature error is the main source that affects the precision of POS. Traditional temperature error model is single temperature parameter linear function, which is not sufficient for the higher accuracy requirement of POS. The traditional compensation method based on neural network faces great problem in the repeatability error under different temperature conditions. In order to improve the precision and generalization ability of the temperature error compensation for POS, a nonlinear multiparameters temperature error modeling and compensation method based on Bayesian regularization neural network was proposed. The temperature error of POS was analyzed and a nonlinear multiparameters model was established. Bayesian regularization method was used as the evaluation criterion, which further optimized the coefficients of the temperature error. The experimental results show that the proposed method can improve temperature environmental adaptability and precision. The developed POS had been successfully applied in airborne TSMFTIS remote sensing system for the first time, which improved the accuracy of the reconstructed spectrum by 47.99%.

1. Introduction

The position and orientation system (POS) is a special strapdown inertial navigation system (SINS)/global positioning system (GPS) integrated measurement equipment [1]. Compared with the traditional SINS/GPS integrated system, it is small, lightweight, and is also available to provide high-precision position, velocity, and attitude information for airborne remote sensing applications including spectrometer (such as Temporally and Spatially Modulated Fourier Transform Imaging Spectrometer (TSMFTIS)), Synthetic Aperture Radar (SAR), Light Detection and Ranging (LiDAR), optical camera, and various other land and marine applications. It has been widely used as a key equipment to further improve imaging quality and efficiency of airborne remote sensing systems [2–4].

However, the precision of POS decreases rapidly with the environmental temperature change [5]. Therefore, it is vital to model and compensate the temperature error of POS. Traditional temperature error modeling and compensation for POS include two kinds of methods: one is polynomial fitting based on least square or multiple regression algorithm to establish the relationship between the temperature input and the POS output [6]. These methods can diminish computational complexity and can be suitable for real-time signal processing of simple model, which are used to compensate temperature error of a traditional SINS/GPS integrated system. However, their precision is limited because

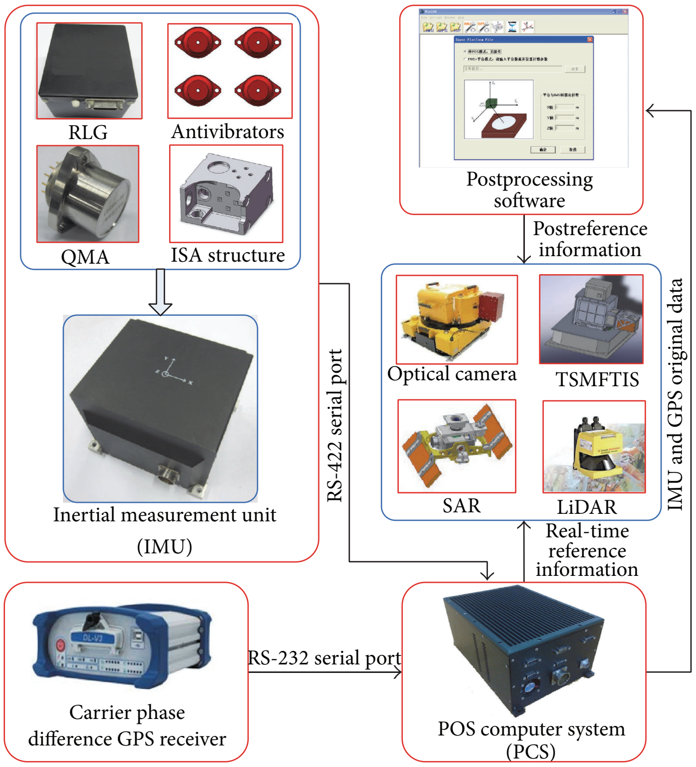

RLG
Antivibrators
QMA
ISA structure
Inertial measurement unit (IMU)
RS-422 serial port
Postprocessing software
Postreference information
Optical camera
TSMFTIS
SAR
LiDAR
IMU and GPS original data
Real-time reference information
Carrier phase difference GPS receiver
RS-232 serial port
POS computer system (PCS)

Figure 1: The components and operation principle of POS.

the model only considers the single temperature without sufficient parameters. The other is based on machine learning technique, such as support vector machine (SVM) [7] and radial basis function (RBF) neural network [8]. In general, RBF neural network is a massively parallel-distributed processor that can be used in complex model with multiparameters and nonlinear problems [9] and can improve the precision in postprocessing of POS. However, traditional RBF neural network method is barely applied to new samples because of its poor generalization ability, which leads to the low repeatability of the compensated results under different temperature condition [10].

Therefore, how to establish a sufficient model and improve the generalization ability has become a key technology for modeling and compensating temperature error of POS. To solve this problem, a nonlinear multiparameters temperature error modeling and compensation method based on Bayesian regularization neural network is proposed in this paper. All the influence factors including temperature, rate of temperature change, and temperature gradient are considered to establish a sufficient multiparameters and nonlinear temperature error model. Bayesian regularization is regarded as evaluation criterion to optimize the coefficients of the temperature error for POS, which further improves the generalization ability. Temperature experiment is implemented to validate the proposed method. The first airborne application experiment of TSMFTIS proves that TSMFTIS has an evident improvement on imaging quality and precision of reconstructed spectrum by using POS data.

This paper is organized as follows. Section 2 introduces the components and operation principle of POS. In Section 3, the temperature error modeling and compensation method of POS based on Bayesian regularization neural network is proposed. In Section 4, the temperature experiment result is presented and validates effectiveness of the proposed method. The airborne experiment of TSMFTIS with POS is introduced in Section 5. Finally, the paper is concluded in Section 6.

2. The Components and Operation Principle of POS

POS is comprised of four main components which are inertial measurement unit (IMU), carrier phase differential GPS receiver, POS computer system (PCS), and postprocessing software as shown in Figure 1.

IMU is mounted directly onto imaging payloads and measures the motion rotation rate and the linear accelerometer information, which determines the precision of POS directly [11]. IMU mainly consists of three ring laser gyroscopes (RLG) and three quartz mechanical accelerometers (QMA) assembled in inertial sensing assembly (ISA) structure in orthogonal triads [12]. The ISA with inertial sensors is installed on exterior supporting frame by eight antivibrators.

A carrier phase differential GPS receiver can provide time, position, and velocity information. PCS is a multifunctional computer system and mainly completes the IMU data sampling and storage, data synchronization, real-time integrated navigation computing, and communication with

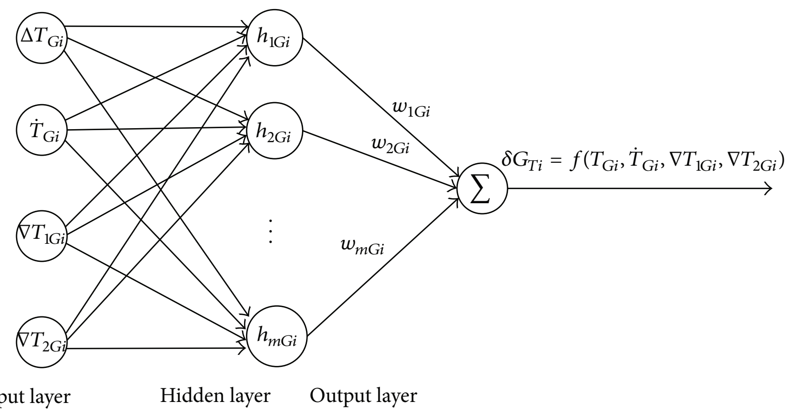


Figure 2: Temperature error modeling for gyroscope assembly based on RBF neural network.

other systems. The integrated system of POS provides excellent short-term dynamics greatly and has none of the long-term drift problem associated with inertial measurement system. The result can be obtained in real time by PCS or in postprocessing by postprocessing software. The postprocessing algorithm in software is available to calculate complex model of massive computation without time limit, which can further improve the precision of POS. As a result, POS can successively provide high-precision position, velocity, and attitude information to improve imaging quality and efficiency for airborne remote sensing system [13].

3. Temperature Error Modeling and Compensation of POS

The output errors of gyroscope assembly in angular rate channel for POS mainly include bias and scale factor error. For RLG, scale factor error varies little with the change of temperature, while bias presents the opposite property [14]. Hence, the temperature error of gyroscope assembly mainly refers to the temperature error of gyroscope bias influenced by the temperature, rate of temperature change, and temperature gradient. The output errors of accelerometer assembly in linear acceleration channel for POS include bias and scale factor error. Temperature changes have a great influence on bias of accelerometers. All these properties mentioned above make it necessary to compensate bias temperature errors of gyroscope and accelerometer assemblies to improve the efficiency and precision of POS.

3.1. Nonlinear Multiparameters Temperature Error Modeling of Gyroscope Assembly. Three RLGs are assembled orthogonally in ISA structure of IMU. The RLG in the ith ($i = x, y, z$) axis embeds three thermometers, which measure the temperature of anode (T_{aGi}), cathode (T_{cGi}), and shell (T_{sGi}), respectively. The temperature, which may result in thermal deformation of different materials in RLG, will induce the change of light path of RLG [8]. The temperature gradient, exaggerated by the rapid change of external temperature, also affects the output of RLG since it leads to extrusion and deflection of components in RLG. All these factors should be considered in the temperature error model of gyroscope assembly for POS.

In order to establish the sufficient temperature error model of gyroscope assembly, the multiple parameters including temperature, temperature change, and temperature gradient should be considered. Therefore, the input $\mathbf{X}_{Gi}$ of gyroscope in the ith axes is given as follows:

$$\mathbf{X}_{Gi} = \begin{bmatrix} T_{Gi} \\ \dot{T}_{Gi} \\ \nabla T_{1Gi} \\ \nabla T_{2Gi} \end{bmatrix} = \begin{bmatrix} \dfrac{T_{aGi} + T_{cGi} + T_{sGi}}{3} \\ \dfrac{\dot{T}_{aGi} + \dot{T}_{cGi} + \dot{T}_{sGi}}{3} \\ T_{aGi} - T_{sGi} \\ T_{cGi} - T_{sGi} \end{bmatrix}, \quad (1)$$

where temperature T_{Gi} is the average of the three temperature values for gyroscope in the ith axes, which has been smoothed to estimate the measurement error. The rate of temperature change $\dot{T}_{Gi}$ is the derivative of T_{Gi}. Limited by the measurement condition, temperature gradients ∇T_{1Gi} and ∇T_{2Gi} are replaced by the temperature differences T_{aGi} and T_{cGi} from T_{sGi}, respectively, because T_{sGi} is the closest to the external environment temperature.

Therefore, the nonlinear multiparameters temperature error model of gyroscope in the ith axes can be expressed qualitatively as the function of the temperature input $\mathbf{X}_{Gi}$:

$$\delta \overline{G}_{Ti} = f(\mathbf{X}_{Gi}) = f\left(\left[T_{Gi}, \dot{T}_{Gi}, \nabla T_{1Gi}, \nabla T_{2Gi}\right]^{T}\right). \quad (2)$$

The RBF neural network is a special feed-forward network and commonly has three layers: an input layer, a single hidden layer, and an output layer [15]. The hidden layer transfers the linear model into a nonlinear one and maps the input space to a new solution space. According to the characteristic of nonlinear and multiparameters in the temperature error model of gyroscope assembly, the RBF neural network here can be given in Figure 2.

Radial basis function is usually defined as a function of the Euclidean distance from arbitrary input $\mathbf{X}_{Gi}$ to

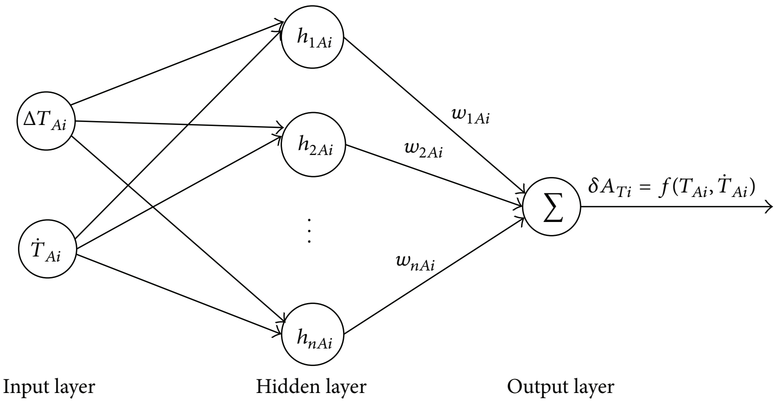


FIGURE 3: Temperature error modeling for accelerometer assembly based on RBF neural network.

the center $\mathbf{C}_{jGi}$; it can be written as h_{jGi}. The most common radial basis function is Gauss kernel function:

$$h_{jGi} = \exp\left\{-\frac{\left\|\mathbf{X}_{Gi} - \mathbf{C}_{jGi}\right\|^2}{2\sigma_{jGi}^2}\right\}, \tag{3}$$

where $\mathbf{C}_{jGi}$ is the basis function center of the jth node in the hidden layer for gyroscope in the ith axes and σ_{jGi} represents the spread of the basis function.

The neural network theory holds that any function can be expressed as a weighted sum of a set of basis functions in solving function approximation problem. So the temperature error model of gyroscope assembly can be given as follows:

$$\begin{aligned} \delta\overline{G}_{Ti} &= f\left(\mathbf{X}_{Gi}\right) = \sum_{j=1}^{M_G} w_{jGi} h_{jGi} \\ &= \sum_{j=1}^{M_G} w_{jGi} \exp\left\{-\frac{\left\|\mathbf{X}_{Gi} - \mathbf{C}_{jGi}\right\|^2}{2\sigma_{jGi}^2}\right\}, \end{aligned} \tag{4}$$

where w_{jGi} is the jth the weight value from the hidden layer to the output layer of gyroscope in the ith axes; M_G represents the node number of the hidden layer. $\mathbf{C}_{jGi}$, σ_{jGi}, and w_{jGi} can be obtained by training process of RBF neural network during the temperature compensation.

3.2. Nonlinear Multiparameters Temperature Error Modeling of Accelerometer Assembly. Unlike RLG, QMA only has one embedded thermometer. The accelerometer assembly temperature error model only can obtain temperature and rate of temperature change without temperature gradient. According to the characteristic of temperature error for accelerometer assembly, the nonlinear multiparameters temperature error model of accelerometer in the ith axes can be expressed qualitatively as the function of the temperature input $\mathbf{X}_{Ai}$:

$$\delta\overline{A}_{Ti} = f\left(\mathbf{X}_{Ai}\right) = f\left(\left[T_{Ai}, \dot{T}_{Ai}\right]^T\right). \tag{5}$$

The temperature input of accelerometer in the ith axes is $\mathbf{X}_{Ai} = [T_{Ai} \;\; \dot{T}_{Ai}]^T$ with T_{Ai} being the output of thermometer embedded in accelerometer in the ith axes; $\dot{T}_{Ai}$ is the derivative of T_{Ai}.

The RBF neural network of accelerometer assembly is given in Figure 3.

The radial basis function is also Gauss kernel function.

The temperature error model of accelerometer assembly can be given as follows:

$$\begin{aligned} \delta\overline{A}_{Ti} &= f\left(\mathbf{X}_{Ai}\right) = \sum_{j=1}^{M_A} w_{jAi} h_{jAi} \\ &= \sum_{j=1}^{M_A} w_{jAi} \exp\left\{-\frac{\left\|\mathbf{X}_{Ai} - \mathbf{C}_{jAi}\right\|^2}{2\sigma_{jAi}^2}\right\}, \end{aligned} \tag{6}$$

where w_{jAi} is the jth the weight value from the hidden layer to the output layer of accelerometer in the ith axes; M_A represents the node number of the hidden layer. $\mathbf{C}_{jAi}$ is the basis function center of the jth node in the hidden layer for accelerometer in the ith axes and σ_{jAi} represents the spread of the basis function. $\mathbf{C}_{jAi}$, σ_{jAi}, and w_{jAi} can be obtained by training process of RBF neural network during the temperature compensation.

3.3. Temperature Error Compensation Method of RBF Neural Network Based on Bayesian Regularization. The poor generalization ability is the most troubling problem affecting the application of RBF neural network. In the traditional algorithm of RBF neural network, the mean square error (MSE) is usually used as the evaluation criterion [14]. The MSE of temperature error in gyroscope and accelerometer assemblies can be given as follows:

$$\begin{aligned} E_{dGi} &= \frac{1}{N}\sum_{n=1}^{N}\left(f\left(\mathbf{X}_{Gi}(n)\right) - \delta\overline{G}_{Ti}(n)\right)^2, \\ E_{dAi} &= \frac{1}{N}\sum_{n=1}^{N}\left(f\left(\mathbf{X}_{Ai}(n)\right) - \delta\overline{A}_{Ti}(n)\right)^2, \end{aligned} \tag{7}$$

where $\delta\overline{G}_{Ti}(n)$ and $\delta\overline{A}_{Ti}(n)$ are the nth training target of the sample for gyroscope and accelerometer in the ith axes, respectively; $X_{Gi}(n)$ and $X_{Ai}(n)$ are the corresponding temperature input for gyroscope and accelerometer in the ith axes, respectively; $f(\mathbf{X}_{Gi}(n))$ and $f(\mathbf{X}_{Ai}(n))$ are the output of the training sample for gyroscope and accelerometer in the ith axes, respectively; N is the total number of the training sample.

The MSE of the training sample can only reflect the ability to approximate the sample of RBF neural network, not representing the generalization ability. A major issue for traditional RBF neural network methods is the potential for overfitting which leads to a fitting of the noise and loses generalization of the network [16]. To reduce the potential for overfitting, an improved RBF neural network based on Bayesian regularization is proposed to prevent the weight values from growing too large by appending weight values in the evaluation criterion. The smaller the weight values are, the better the generalization capability of the network is. The weight decay regularizer can be given as follows:

$$\begin{aligned} E_{wGi} &= \frac{1}{M_G}\sum_{k=1}^{M_G} w_{jGi}^2, \\ E_{wAi} &= \frac{1}{M_A}\sum_{k=1}^{M_A} w_{jAi}^2. \end{aligned} \tag{8}$$

So the evaluation criterion of Bayesian regularization can be given as follows:

$$\begin{aligned} F_{Gi}\left(w_{Gi}\right) &= \alpha_{Gi}E_{wGi} + \beta_{Gi}E_{dGi}, \\ F_{Ai}\left(w_{Ai}\right) &= \alpha_{Ai}E_{wAi} + \beta_{Ai}E_{dAi}, \end{aligned} \tag{9}$$

where α_{Gi}, β_{Gi}, α_{Ai}, and β_{Ai} are the evaluation criterion function parameters. Under the situation of $\alpha_{Gi} \ll \beta_{Gi}$ or $\alpha_{Ai} \ll \beta_{Ai}$, overfitting may occur due to the overemphasis on reducing training error. While on the other hand, too much concentration on limiting the network weight values when $\alpha_{Gi} \gg \beta_{Gi}$ or $\alpha_{Ai} \gg \beta_{Ai}$ may probably lead to large training error. Therefore, how to find optimal values of α_{Gi}, β_{Gi}, α_{Ai}, and β_{Ai} is a key point of Bayesian regularization. The Bayesian regularization to RBF network training makes use of an iterative procedure in the manner of the expectation-maximization algorithm for estimating the network weights. The iterations consist of finding the most probable value for the weights from their distribution [17]. In the Bayesian framework, the weight values of the network are assumed to be random variables. According to Bayesian's rule [18], the optimization of the regularization parameters α_{Gi}, β_{Gi}, α_{Ai}, and β_{Ai} require solving the Hessian matrix of $F_{Gi}(w_{Gi}^{MP})$ and $F_{Ai}(w_{Ai}^{MP})$ at the minimum point w_{Gi}^{MP} and w_{Ai}^{MP}. The optimal weight values w_{Gi}^{MP} and w_{Ai}^{MP} should maximize the posterior probability; this is equivalent to minimizing the regularized evaluation criterion functions $F_{Gi}(w_{Gi})$ and $F_{Ai}(w_{Ai})$.

Initializing the value of $\alpha_{Gi}(0)$, $\beta_{Gi}(0)$, $\alpha_{Ai}(0)$, and $\beta_{Ai}(0)$ by prior distribution, the updating formula of the evaluation criterion function parameters in gyroscope and accelerometer assemblies can be expressed as follows:

$$\begin{aligned} H_{Gi}(k) &= \beta_{Gi}(k)\nabla^2 E_{dGi}\left(w_{Gi}^{MP}(k)\right) \\ &\quad + \alpha_{Gi}(k)\nabla^2 E_{wGi}\left(w_{Gi}^{MP}(k)\right), \\ \gamma_{Gi}(k) &= M_G - 2\alpha_{Gi}(k)\operatorname{tr}\left(H_{Gi}(k)\right)^{-1}, \\ \alpha_{Gi}(k+1) &= \frac{\gamma_{Gi}(k)}{2E_{wGi}\left(w_{Gi}^{MP}(k)\right)}, \\ \beta_{Gi}(k+1) &= \frac{N-\gamma_{Gi}(k)}{2E_{dGi}\left(w_{Gi}^{MP}(k)\right)}, \\ H_{Ai}(k) &= \beta_{Ai}(k)\nabla^2 E_{DAi}\left(w_{Ai}^{MP}(k)\right) \\ &\quad + \alpha_{Ai}(k)\nabla^2 E_{wAi}\left(w_{Ai}^{MP}(k)\right), \\ \gamma_{Ai}(k) &= M_A - 2\alpha_{Ai}(k)\operatorname{tr}\left(H_{Ai}(k)\right)^{-1}, \\ \alpha_{Ai}(k+1) &= \frac{\gamma_{Ai}(k)}{2E_{wAi}\left(w_{Ai}^{MP}(k)\right)}, \\ \beta_{Ai}(k+1) &= \frac{N-\gamma_{Ai}(k)}{2E_{dAi}\left(w_{Ai}^{MP}(k)\right)}, \end{aligned} \tag{10}$$

where $\gamma_{Gi}(k)$ and $\gamma_{Ai}(k)$ are called the effective number of parameters. $H_{Gi}(k)$ and $H_{Ai}(k)$ are the Hessian matrix of the evaluation criterion function.

The training process of Bayesian regularization neural network is iterative algorithm for compensating temperature error of POS. The limited function is

$$\begin{aligned} \left|\alpha_{Gi}(k+1) - \alpha_{Gi}(k)\right| &\le e_{\alpha Gi}, \\ \left|\beta_{Gi}(k+1) - \beta_{Gi}(k)\right| &\le e_{\beta Gi}, \\ \left|\alpha_{Ai}(k+1) - \alpha_{Ai}(k)\right| &\le e_{\alpha Ai}, \\ \left|\beta_{Ai}(k+1) - \beta_{Ai}(k)\right| &\le e_{\beta Ai}, \end{aligned} \tag{11}$$

where $e_{\alpha Gi}$, $e_{\beta Gi}$, $e_{\alpha Ai}$, and $e_{\beta Ai}$ are infinitesimals; this is equivalent to the convergence of the evaluation criterion function. When the training process of Bayesian regularization neural network is finished, the optimization of the neural network parameters w_{jGi}^*, $\mathbf{C}_{jGi}^*$, σ_{jGi}^* and w_{jAi}^*, $\mathbf{C}_{jAi}^*$, σ_{jAi}^* can be obtained. The final result of temperature errors $\delta\overline{G}_{Ti}^*$ and $\delta\overline{A}_{Ti}^*$ can be updated by w_{jGi}^*, $\mathbf{C}_{jGi}^*$, σ_{jGi}^* and w_{jAi}^*, $\mathbf{C}_{jAi}^*$, σ_{jAi}^*. The temperature error $\delta\overline{G}_{Ti}^*$ is used to compensate the original output of RLG bias G_{oi} in the ith axes. The temperature error $\delta\overline{A}_{Ti}^*$ is used to compensate the original

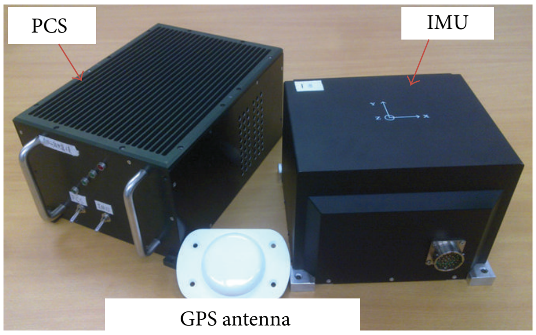


FIGURE 4: The hardware components of developed POS.

output of QMA bias $\overline{A}_{oi}$ in the ith axes. The final output of gyroscope and accelerometer in the ith axes can be given by

$$\begin{aligned} G_{bi} &= G_{oi} - \delta\overline{G}_{Ti} \\ &= G_{oi} - \left(\sum_{j=1}^{M_G} w^*_{jGi} \exp\left\{ -\frac{\left\| \mathbf{X}_{Gi} - \mathbf{C}^*_{jGi} \right\|^2}{2\left(\sigma^*_{jGi}\right)^2} \right\} \right), \\ A_{bi} &= A_{oi} - \delta\overline{A}_{Ti} \\ &= A_{oi} - \left(\sum_{j=1}^{M_A} w^*_{jAi} \exp\left\{ -\frac{\left\| \mathbf{X}_{Ai} - \mathbf{C}^*_{jAi} \right\|^2}{2\left(\sigma^*_{jAi}\right)^2} \right\} \right). \end{aligned} \tag{12}$$

Bayesian regularization method reduces the network weight values under the condition of minimizing network training error by new evaluation criterion which eventually improve the generalization ability of neural network.

4. Temperature Experiment

4.1. The Equipment in Temperature Experiment. In temperature experiment, we developed the POS (Model TX-R20) which is applied in airborne remote sensing system. The POS as shown in Figure 4 can implement gyros biases calibration, initial alignment, inertial navigation algorithms, and so forth. The POS uses an IMU that is sufficient and separated from the PCS and connected to the PCS by a data interface and power cable. The IMU is designed to be small (210 mm × 200 mm × 142 mm), light (6.7 Kg), and of little power consumption (20 W). The common output frequency is 100 Hz (highest being 200 Hz). The overall power consumption of POS including IMU and PCS is 28 W, which is commensurate in performance with the POS/AV610. The nominal accuracy of three RLGs (X, Y, and Z gyroscopes) is less than 0.01°/h and the nominal accuracy of three QMAs (X, Y, and Z accelerometers) is less than 50 μg.

4.2. The Procedures of the Temperature Experiment. IMU is attached in a temperature chamber as shown in Figure 5. The IMU and temperature chamber are required to be stationary during the whole experiment. The temperature of the chamber varies from −30°C to 50°C. To establish the temperature error model of POS, an experiment is implemented.

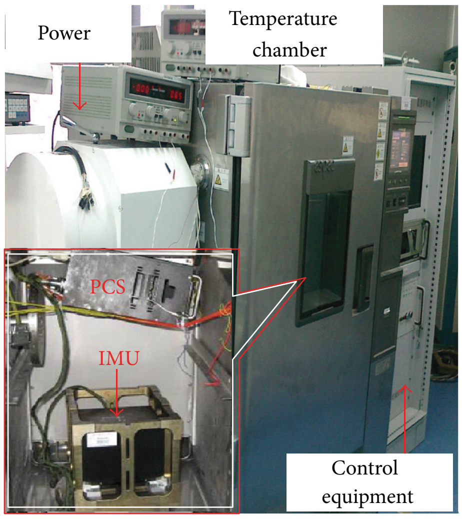


FIGURE 5: The equipment of the temperature experiment.

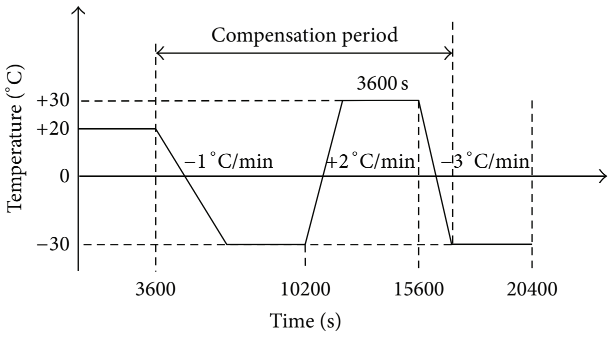


FIGURE 6: The procedures of the temperature experiment.

Firstly, the temperature is kept at 20°C for 1.5 hours to attain thermodynamic equilibrium of RLG and QMA assemblies. Secondly, the temperature of the chamber increases to 50°C with uniform rate of 3°C/min. Thirdly, the temperature is kept at 50°C for 2 hours. Finally, the temperature of the chamber decreases to −30°C with a uniform rate of −3°C/min; then the temperature is also kept for 2 hours.

The temperature errors are chosen as the training target of the RBF neural network, and then it is trained by the traditional RBF neural network and proposed method.

4.3. Compensation Results of Temperature Errors. In order to validate the generalization ability of the RBF neural network trained by the proposed method, a verification test is implemented as shown in Figure 6. Firstly, the temperature is kept at 20°C for 1 hour. Secondly, the temperature of the chamber decreases to −30°C with uniform rate of −1°C/min. The temperature is kept at −30°C for 1 hour. Thirdly, the temperature of the chamber increases to 30°C with a uniform rate of 2°C/min. The temperature is kept at 30°C for 1 hour. Finally, the temperature of the chamber decreases to −30°C with a uniform rate of −3°C/min and is kept at −30°C for 1 hour.

Table 1: The compensational results in gyroscope assembly.

Gyroscope	Original bias (°/h)	Traditional method (°/h)	Proposed method (°/h)	Improvement (%)
X	0.0298	0.0087	0.0078	10.4
Y	0.0130	0.0104	0.0098	5.7
Z	0.0232	0.0091	0.0080	12.1

Table 2: The compensational results in accelerometer assembly.

Accelerometer	Original bias (μg)	Traditional method (μg)	Proposed method (μg)	Improvement (%)
X	181.37	24.52	18.96	22.6
Y	199.07	30.33	21.89	27.8
Z	438.58	24.13	16.58	31.3

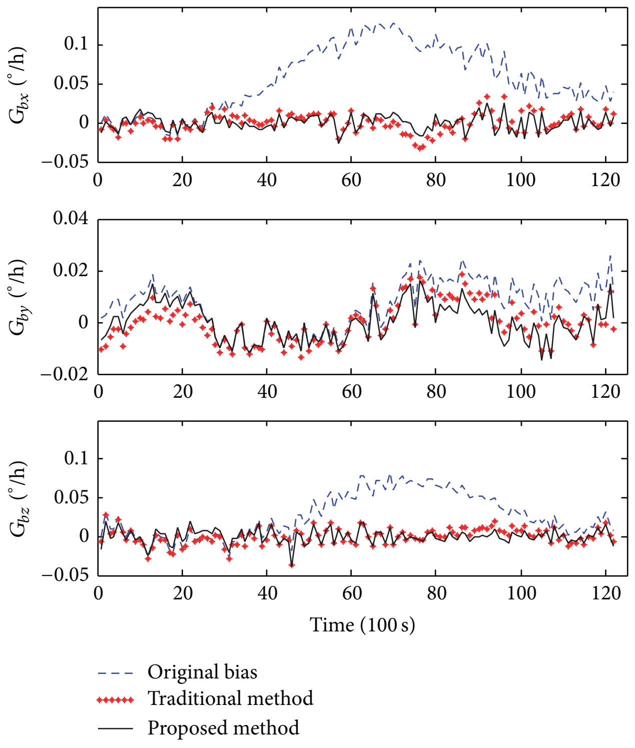


Figure 7: Curves of RLG bias changing with temperature.

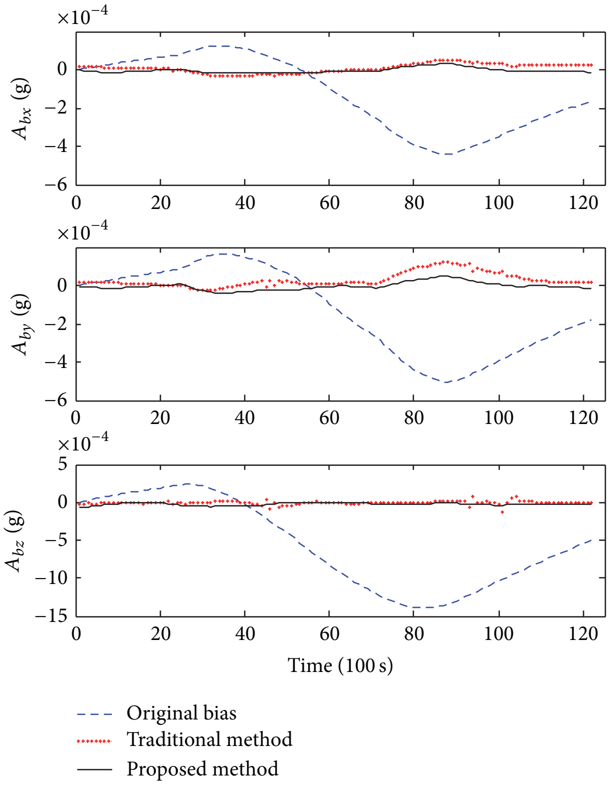


Figure 8: Curves of QMA bias changing with temperature.

The temperature errors of gyroscope and accelerometer assemblies in verification test are compensated by the traditional RBF neural network and proposed method in the paper. The compensation results of gyroscope and accelerometer assemblies are shown in Figures 7 and 8. The comparisons of results are given in Tables 1 and 2.

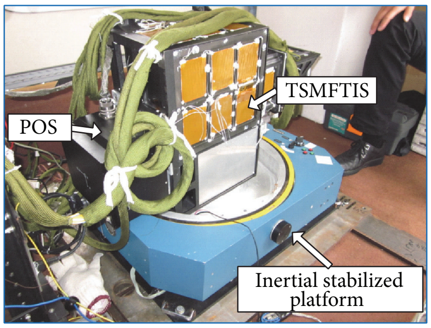
As it is evident from Tables 1 and 2, the proposed method reduces the temperature errors of RLG bias and QMA bias effectively.

5. The Airborne Application Experiment

In order to further validate the application capability of the proposed method, the airborne application experiment shown in Figure 9 is implemented in July 2013, Weihai City, Shandong, China. The equipment of the airborne application experiments includes an experimental airplane (Y-12), TSMFTIS, and POS. The POS and TSMFTIS are fastened together and then installed on the inertial stabilized platform. The inertial stabilized platform is connected with the cabin of the airplane. Based on the motion information provided by POS, TSMFTIS successfully accomplishes airborne remote sensing assignment.

5.1. The Result of Airborne Experiment. The airplane has a flight of 3 hours and 20 minutes and flies at a height of 2800 meters. The environment temperature changes rapidly throughout the flight. During the 15-minute climbing, the external temperature changes rapidly from 34°C to 19°C,

The experimental aircraft applied in remote sensing



The POS and TSMFTIS in the aircraft

FIGURE 9: The airborne application experiments.

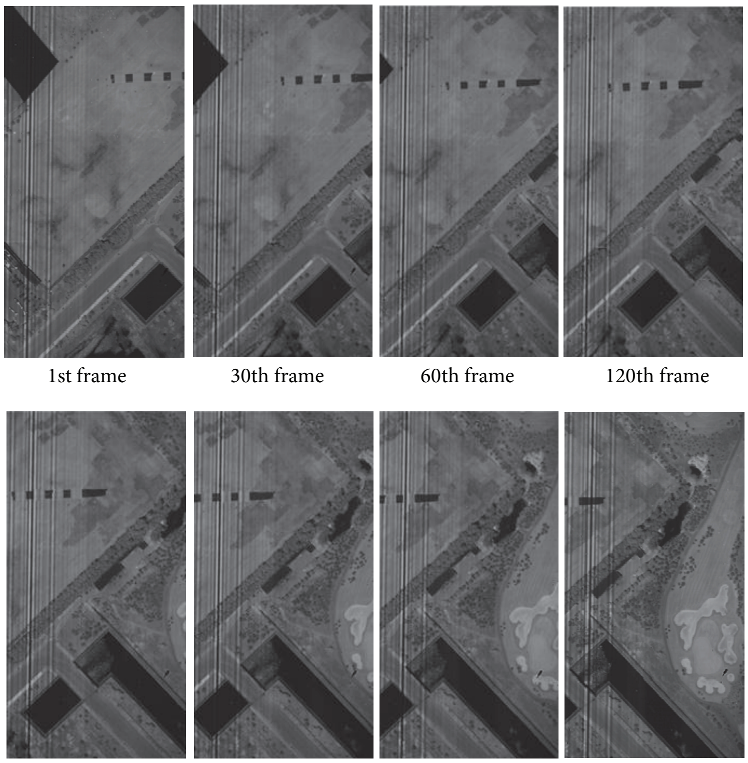
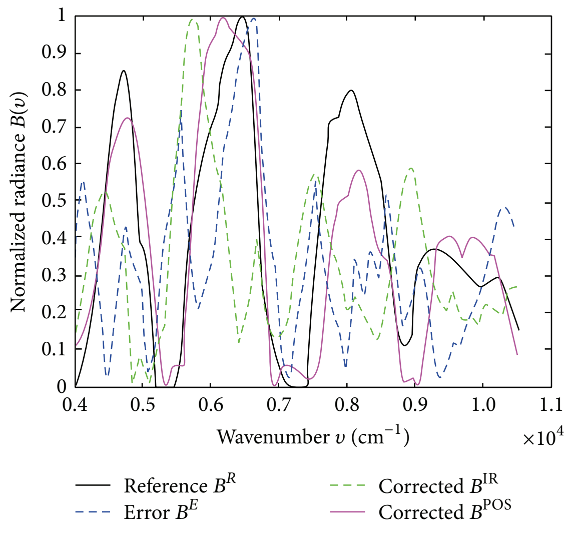


FIGURE 10: The interferogram with TSMFTIS using POS data.

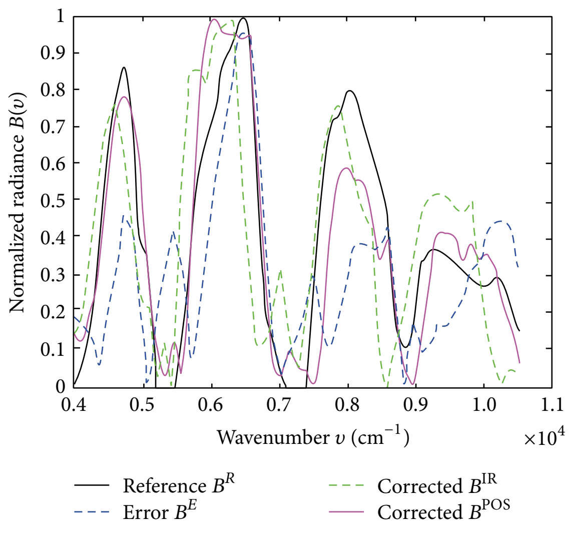
which affects the measurement accuracy of inertial navigation. Accordingly, temperature error compensation is necessary in inertial navigation. The inertial navigation error results by traditional RBF neural network method and the proposed method are shown in Table 3.

The results show that compared to the traditional RBF neural network method, the inertial navigation error of proposed method is reduced to 0.57 nautical mile from 0.63 nautical mile during 1 hour, falling by 9.52%. The inertial navigation error over 3 hours and 20 minutes is reduced to 1.60 nautical miles from 1.73 nautical miles, falling by 7.51%.

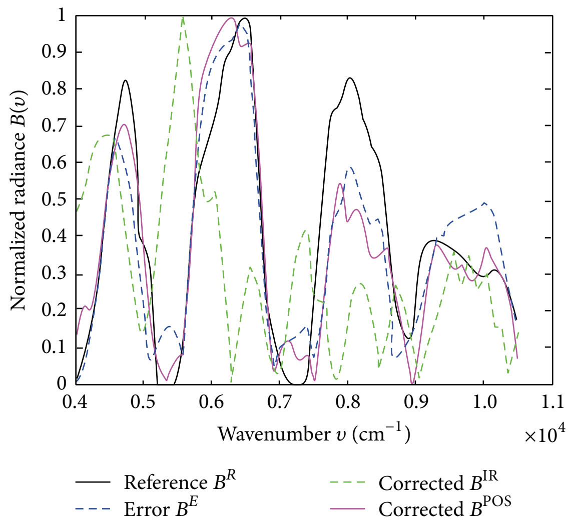
5.2. The Airborne Remote Sensing Experiment of TSMFTIS with POS. The application of TSMFTIS, which plays a vital role in the remote sensing system, has no internal moving parts and high throughput advantages [19]. It is usually applied in satellite remote sensing assignment which requires high stability of the platform to provide accurate position, velocity, and attitude references for TSMFTIS to measure the geographic position and orientation directly during the push-broom process [20]. However, in airborne remote sensing system, the attitude changes of the aircraft due to air disturbance cannot be completely compensated



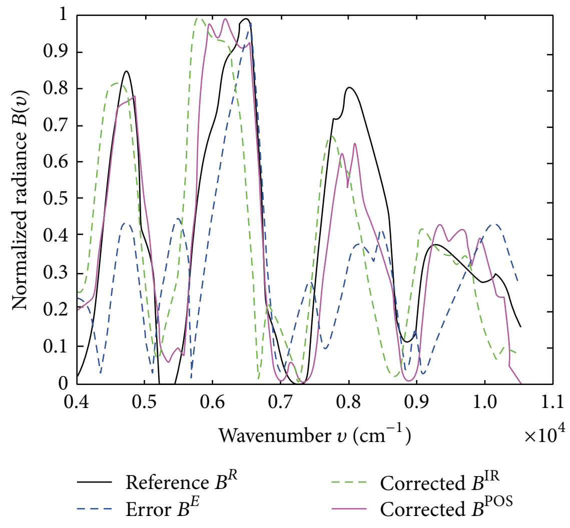
(a)



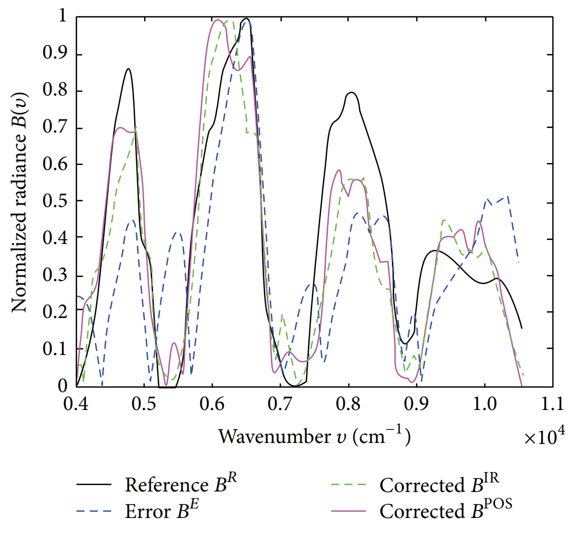
(b)



(c)



(d)



(e)

Figure 11: Five groups of the reconstructed spectra.

Table 3: Inertial navigation error in the airborne experiments.

	Traditional RBFNN	Proposed method	Improvement in precision
1 hour of navigation error (nautical mile)	0.63	0.57	9.52%
3 hours 20 minutes of navigation error (nautical mile)	1.73	1.60	7.51%

Table 4: The RQE results in airborne remote sensing experiment.

Group	Traditional method	Image registration	Target-tracking-based	Improvement (%)
1	0.6410	0.7275	0.3141	51.00
2	0.4930	0.4407	0.2776	43.69
3	0.6450	0.3474	0.2608	59.56
4	0.4749	0.4799	0.2637	44.47
5	0.4697	0.2893	0.2761	41.22
Average	0.5447	0.4569	0.2784	47.99

by the inertial stabilized platform because of its limit on stability and precision which cannot completely compensate the deviations. Therefore, the aircraft cannot be regarded as a stable platform compared with satellites platforms, which restricts the application of TSMFTIS in airborne remote sensing; sometimes even the interferogram cannot be acquired. This experiment, as an innovative attempt of the application of TSMFTIS on airborne remote sensing system, efficiently achieves target-tracking-based method, which considers POS data as its vital element [21].

POS data provides attitude information on TSMFTIS in remote sensing imaging, which as a result tracks the position of the target on each recorded image and obtains the right target interferogram for airborne TSMFTIS. The right target interferogram by target-tracking-based method using POS data, which is obtained from the proposed method, is shown in Figure 10. However, the other methods cannot obtain all the effective interferograms without POS data.

In Figure 11, the reconstructed spectrum from the reference interferograms by spectroradiometer is denoted by reference spectrum B^R, while B^E is the error spectrum reconstructed from distorted interferogram. B^{IR} and B^{POS} represent the spectra reconstructed from the corrected interferogram based on image registration method and target-tracking-based method obtained by POS data, respectively [22, 23]. The reconstructed spectrum using POS data matches the reference spectrum better than that from image registration method.

The relative spectral quadratic error (RQE) [24] result shown in Table 4 is regarded as an evaluation standard of spectral precision. Table 4 demonstrates that the accuracy of the reconstructed spectrum by target-tracking-based method using POS data improves rapidly compared with traditional method in all five experiments, which validates both the effectiveness and robustness of the proposed method. The average result shows that the RQE is reduced to 0.2784 from 0.5447, falling by 47.99%.

6. Conclusion

POS plays a vital role and becomes a research and development focus in airborne remote sensing systems. In this paper, a nonlinear multiparameters temperature error model and compensation method of POS based on Bayesian regularization neural network were proposed to improve the measurement precision of POS. At first, the nonlinear multiparameters temperature error model of gyroscope and accelerometer assemblies of POS was established. Then, a Bayesian regularization neural network method was proposed to improve the generalization ability of the network, which could be used as the evaluation criterion to optimize the configuration of temperature error. The experiment results show that the compensated temperature errors of gyroscope and accelerometer assemblies are reduced compared to the traditional method. The proposed method had been applied in postprocessing algorithm of POS, which successfully accomplished the first airborne TSMFTIS remote sensing assignment.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

This work was supported in part by the National Program on Key Basic Research Projects of China under Grant 2009CB724002, the CAST Innovation Foundation CAST201205, the Beijing Youth Elite Project, the Basic Scientific Research YWF-14-YQGD-001, the National Special Instrument Project 2012YQ160185, the Science and Technology Project of China Southern Grid 20100002012117034, and in part by the National Natural Science Foundation of China under Grants 60825305, 61121003, and 61104198.

References

[1] J. C. Fang and S. Yang, "Study on innovation adaptive EKF for in-flight alignment of airborne POS," *IEEE Transactions on Instrumentation and Measurement*, vol. 60, no. 4, pp. 1378–1388, 2011.

[2] M. M. R. Mostafa and K.-P. Schwarz, "Digital image georeferencing from a multiple camera system by GPS/INS," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 56, no. 1, pp. 1–12, 2001.

[3] J. Li, J. Fang, and M. Du, "Error analysis and gyro-bias calibration of analytic coarse alignment for airborne POS," *IEEE Transactions on Instrumentation and Measurement*, vol. 61, no. 11, pp. 3058–3064, 2012.

[4] H.-S. Ahn and C.-H. Won, "DGPS/IMU integration-based geolocation system: Airborne experimental test results," *Aerospace Science and Technology*, vol. 13, no. 6, pp. 316–324, 2009.

[5] J. Cheng and J. Fang, "Comparison of compensation methods on RLG temperature error and their application in POS," in *Proceedings of the 8th IEEE International Symposium on Instrumentation and Control Technology (ISICT '12)*, pp. 189–194, July 2012.

[6] C. Guo, Y. Xu, and X. Zhao, "Investigation on the temperature compensating model ring laser gyroscope," *Chinese Optics Letters*, vol. 4, no. 10, pp. 576–579, 2006.

[7] G. Wei, G. Li, Y. Wu, and X. Long, "Application of Least Squares-Support Vector Machine in system-level temperature compensation of ring laser gyroscope," *Measurement*, vol. 44, no. 10, pp. 1898–1903, 2011.

[8] Q. Zhang, X.-F. Liu, J. Zhan, and G.-M. Chen, "Temperature modeling study for high precision gyroscope based on neural network," in *Proceedings of the International Symposium on Intelligent Ubiquitous Computing and Education (IUCE '09)*, pp. 85–87, May 2009.

[9] R. Sharaf and A. Noureldin, "A neural network model of optical gyros drift errors with application to vehicular navigation," in *Applications of Digital Image Processing XXVII*, vol. 5558 of *Proceedings of SPIE*, pp. 13–20, August 2004.

[10] Q. Hua, Y. Gao, X.-Z. Wang, and B.-Y. Zhao, "A new approach to improving generalization ability of feed-foward neural networks," in *Proceedings of the International Conference on Machine Learning and Cybernetics (ICMLC '10)*, pp. 1413–1419, July 2010.

[11] J. Li, A. Chen, J. Fang, and J. Cheng, "Time delay modeling and compensation of dithered RLG POS with antivibrators and filter," *Measurement*, vol. 46, no. 6, pp. 1928–1937, 2013.

[12] J. Li, J. Fang, and S. S. Ge, "Kinetics and design of a mechanically dithered ring laser gyroscope position and orientation system," *IEEE Transactions on Instrumentation and Measurement*, vol. 62, no. 1, pp. 210–220, 2013.

[13] E. Lithopoulos, B. Reid, and B. Scherzinger, "The position and orientation system (POS) for survey applications," in *Proceedings of the International Archives of Photogrammetry and Remote Sensing*, pp. 467–471, 1996.

[14] J. S. Cain, D. A. Staley, G. R. Heppler, and J. McPhee, "Stability analysis of a dynamically timed gyroscope," *Journal of Guidance, Control, and Dynamics*, vol. 29, no. 4, pp. 965–969, 2006.

[15] L. Franco, J. M. Jerez, and J. M. Bravo, "Role of function complexity and network size in the generalization ability of feedforward networks," in *Proceedings of the 8th International Workshop on Artificial Neural Networks (IWANN '05)*, pp. 1–8, June 2005.

[16] J. L. Ticknor, "A Bayesian regularized artificial neural network for stock market forecasting," *Expert Systems with Applications*, vol. 40, no. 14, pp. 5501–5506, 2013.

[17] E. Rank, "Application of Bayesian trained RBF networks to nonlinear time-series modeling," *Signal Processing*, vol. 83, no. 7, pp. 1393–1410, 2003.

[18] P. Kumar, S. N. Merchant, and U. B. Desai, "Improving performance in pulse radar detection using Bayesian regularization for neural network training," *Digital Signal Processing*, vol. 14, no. 5, pp. 438–448, 2004.

[19] L. J. Su, Y. Yuan et al., "Spectrum reconstruction method for airborne temporally-spatially modulated Fourier transform imaging spectrometers," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 52, no. 6, pp. 3720–3728, 2014.

[20] J. Reguera-Salgado, M. Calvino-Cancela, and J. Martin-Herrero, "GPU geocorrection for airborne pushbroom imagers," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 50, no. 11, pp. 4409–4419, 2012.

[21] Y. Yuan, X. Zhang, C. Sun, and Z. Zhou, "Modeling of the temporally and spatially modulated fourier transform imaging spectrometer working in orbit," *Optik*, vol. 122, no. 17, pp. 1576–1583, 2011.

[22] X. Fan, H. Rhody, and E. Saber, "A spatial-feature-enhanced MMI algorithm for multimodal airborne image registration," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 48, no. 6, pp. 2580–2589, 2010.

[23] P. Vandewalle, S. Süsstrunk, and M. Vetterll, "A frequency domain approach to registration of aliased images with application to super-resolution," *EURASIP Journal on Applied Signal Processing*, vol. 2006, Article ID 71459, 14 pages, 2006.

[24] C. Mailhes, P. Vermande, and F. Castanié, "Spectral image compression," *Journal of Optics*, vol. 21, no. 3, pp. 121–132, 1990.

Hindawi Publishing Corporation
Journal of Applied Mathematics
Volume 2014, Article ID 640406, 8 pages
http://dx.doi.org/10.1155/2014/640406

Research Article

The Application of Pattern Recognition in Electrofacies Analysis

Huan Li,[1] Xiao Yang,[2] and Wenhong Wei[1]

[1] *Dongguan University of Technology, Dongguan 523808, China*
[2] *School of Information Science and Technology, Tsinghua University, Beijing 100084, China*

Correspondence should be addressed to Wenhong Wei; weiwh@dgut.edu.cn

Received 24 February 2014; Accepted 26 April 2014; Published 4 June 2014

Academic Editor: Guiming Luo

Pattern recognition is an important analytical tool in electrofacies analysis. In this paper, we study several commonly used clustering and classification algorithms. On the basis of advantages and disadvantages of existing algorithms, we introduce the KMRIC algorithm, which improves initial centers of K-means. Also, we propose the AKM algorithm which automatically determines the number of clusters and apply support vector machine to classification. Finally, we apply these algorithms to electrofacies analysis, where the experiments on the real-world datasets are carried out to compare the merits of various algorithms.

1. Introduction

The basic principle of electrofacies analysis is to determine the lithological types corresponding to electrofacies according to the known lithological and underlying parameters in the key well. Then we conduct clustering and discriminant analysis of key well and noncoring wells to automatically judge the automatica.

Clustering means the process of partitioning an unlabeled dataset into groups of similar objects. Each group, called a cluster, consists of objects that are similar to each other with respect to a certain similarity measure and which are dissimilar to objects of other groups. The applications of cluster analysis have been used in a wide range of different areas, including artificial intelligence, bioinformatics, biology, computer vision, data compression, image analysis, information retrieval, machine learning, marketing, medicine, pattern recognition, spatial database analysis, statistics, recommendation systems, and web mining.

Dong et al. [1] proposed an improvement method based on K-means, which obtains the optimized initial center from a group of initial clustering centers. The K-means algorithm is one of the most popular and widespread partitioning clustering algorithms because of its superior feasibility and efficiency in dealing with a large amount of data. The main drawback of the KM algorithm is that the cluster result is sensitive to the selection of the initial cluster centers and may converge to the local optima. At present, the development tendency of clustering method is to find a global optimal solution in combination with the global optimization methods like simulated annealing, particle swarm, and other local methods like K-means [2–6]. Pelleg and Moore [7] proposed an algorithm which can automatically determine the optimal number of clusters during clustering. The challenge of clustering high-dimensional data has emerged in recent years. Clustering high-dimensional data is the cluster analysis of data anywhere from a few dozens to many thousands of dimensions. Such high-dimensional data spaces are often encountered in areas such as medicine, biology, bioinformatics, and the clustering of text documents, where, if a word-frequency vector is used, the number of dimensions equals the size of the dictionary. In high-dimensional clustering, generally the original space is transformed by PCA, SVD, K-L transformation, and other dimensionality reduction methods first; then the clustering of low-dimensional space is performed. Bertini et al. [8] introduced a high-dimensional visualization technology, showing multidimensional data on two-dimensional plane.

K-means [9, 10] is a clustering method most widely used in science and engineering nowadays. However, it has the following 5 deficiencies [3, 5].

(1) The results are initial center initiative.

(2) Only local optimal solution can be obtained, rather than global optimal solution.

(3) The number of clustering k should be set in advanced artificially.

(4) The error point imposes serious impacts on the results of clustering.

(5) The algorithm lacks scalability.

The paper introduces an improved algorithm according to the deficiencies of K-means.

2. Improve K-Means Method of Initial Center

Aimed at the disadvantages (1) and (4) in K-means algorithm, we propose a K-means algorithm with refined initial centers (KMRIC for short) based on the works of predecessors [1].

(1) Randomly extract J sample subsets S_i, $i = 1, 2, \ldots, J$.

(2) Conduct K-means clustering of J sample subsets, respectively, on the whole data field to get J sets CM_i, $i = 1, 2, \ldots, J$, $\mathrm{CM} = \bigcup_{i=1}^{J} \mathrm{CM}_i$, in which there are $K \times J$ points for CM at most.

(3) Conduct K-means clustering on CM by taking CM as the initial clustering center to get J clustering center sets FM_i $i = 1, 2, \ldots, J$.

It can be seen from Figure 1 that the clustering center is obtained from different subsample set, near the real clustering center, and clustering is formed by different subsample set. In (3), selecting the one with the minimum sum of squares of deviations as the improved initial clustering center can reduce the randomness brought by random selection. In (2), to eliminate the influence of error point, the modified K-means algorithm (KmeansMod) is adopted. KmeansMod has the following modification based on the standard K-means: when the standard K-means algorithm is completed, the data point contained in each clustering will be checked. If the data point contained in a clustering is zero, the original center will be replaced by taking the data point furthest to the clustering center as a new center and then the K-means algorithm is reran.

KMRPIC algorithm eliminates the sensitivity of K-means algorithm to data input consequence and initial centers, which is an obvious improvement compared with K-means effect. When applied to large-scale data, KMRIC can reduce the iterations and improve the execution efficiency.

3. Adaptive K-Means

The number of clusters k of K-means algorithm should be set in advance manually. However, actually we do not know the value of k, especially in the case of high dimension of data, so it is more difficult to select the correct value of k.

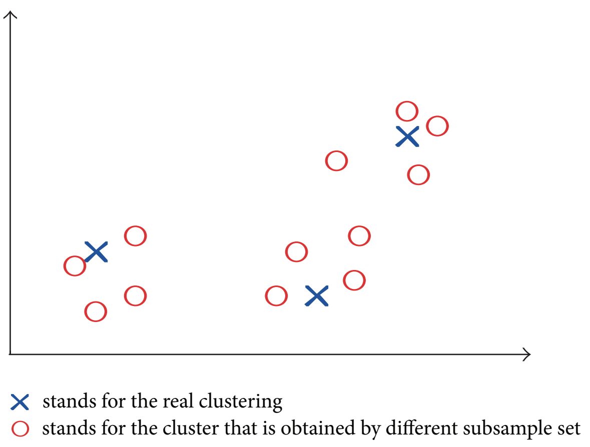

× stands for the real clustering
○ stands for the cluster that is obtained by different subsample set

FIGURE 1: Multicombination clustering center obtained from multiple sample subsets.

X-means put forward by Pelleg and Moore [7] can automatically determine the number of clusters. However, X-means is prone to split data into more clusters than the actual ones, which is particularly obvious when the data is not strictly subject to the normal distribution. Lewis [11] statistics are introduced as the standard of measuring the normal distribution and propose an adaptive K-means (AKM for short).

The AKM algorithm first assumes that all data are in the same cluster; then the number of clustering is gradually increased in the subsequent iterations. In each iteration, whether each cluster satisfied the normal distribution is judged at once; if not, the cluster should be split into two clusters. After each splitting, K-means clustering is carried out in the whole data field to improve the clustering results. The iteration ends until there is no splitting and then the final clustering results will be obtained. The schematic diagram of AKM algorithm is shown in Figure 2. In Figure 2, clustering is divided into three categories firstly; then each category is split into two subclasses. At last, the results are got after one splitting to judge whether each subclass follows Gaussian distribution.

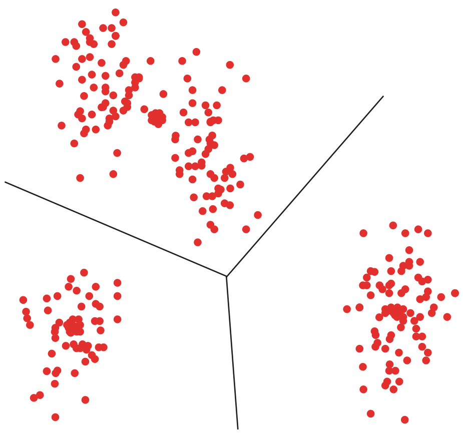
The judgment of splitting is as follows.

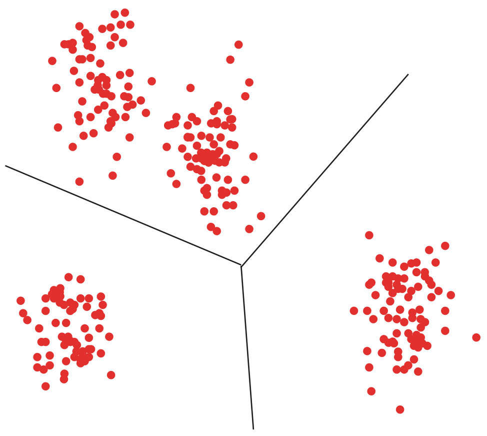
(1) Select the confidence level α.

(2) Run KMRIC program and split X into two to get two clustering centers c_1, c_2.

(3) Let $v = c_1 - c_2$ be an N-dimensional vector connecting the two centers, which is the main direction of judging the normal distribution. X is projected on v: $x_i' = (\langle x_i, v \rangle / \|v\|^2) X'$ is transformed to make its mean as 0 and variance as 1.

(4) Suppose that $z_i = F(x_{(i)}')$. The results $A_*^2(Z)$ with respect to confidence level α are not significant, so accept H_0; reserve the original clustering center c and abandon c_1 and c_2. Otherwise, reject H_0, and replace the original clustering center c by c_1 and c_2.


(a) It is divided into three categories


(b) Each category is split into two subclasses

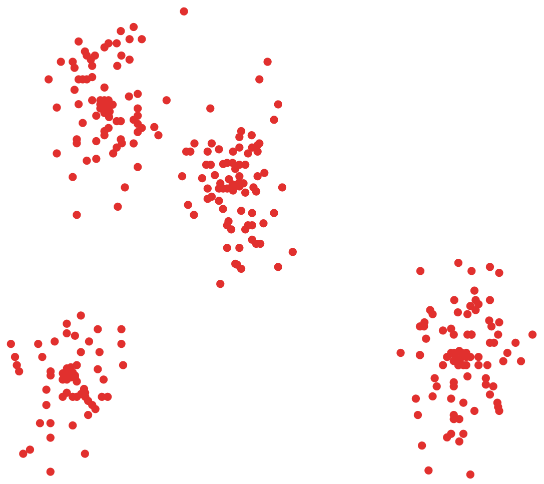

(c) Get the results after one splitting to judge whether each subclass follows Gaussian distribution

Figure 2: Schematic diagram of AKM algorithm.

$A^2_*(Z)$ is the statistics of Anderson Darling:

$$A^2(Z) = -\frac{1}{n}\sum_{i=1}^{n}(2i-1)\left[\log(z_i) + \log(1 - z_{n+1-i})\right] - n. \tag{1}$$

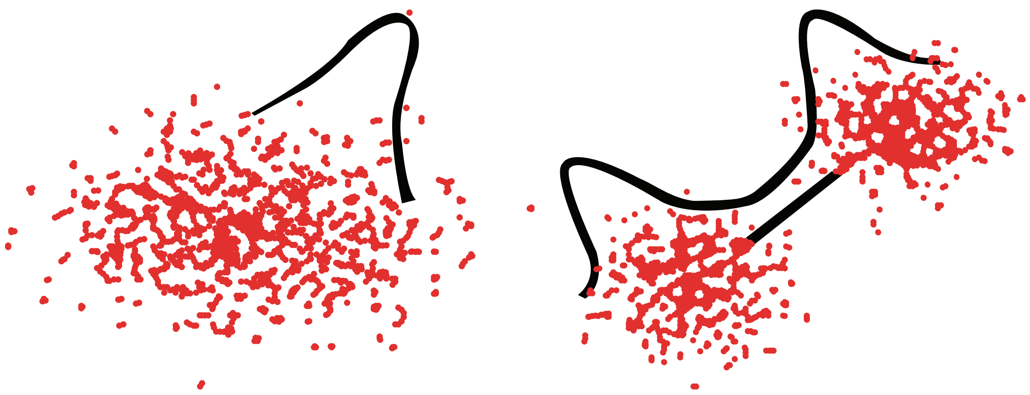
Figure 3 shows two distribution circumstances. In Figure 3(a), the subclass follows Gaussian distribution, but in Figure 3(b), the subclass does not follow Gaussian distribution. AKM algorithm can judge whether each subclass follows Gaussian distribution.

AKM integrates the determination process of the number of clusters and the clustering process, which can automatically determine the optimal number of clusters, thus avoiding the subjectivity in the selection of number of clusters and the blindness of initialization, and can also distinguish the errors.

4. Discriminant Method

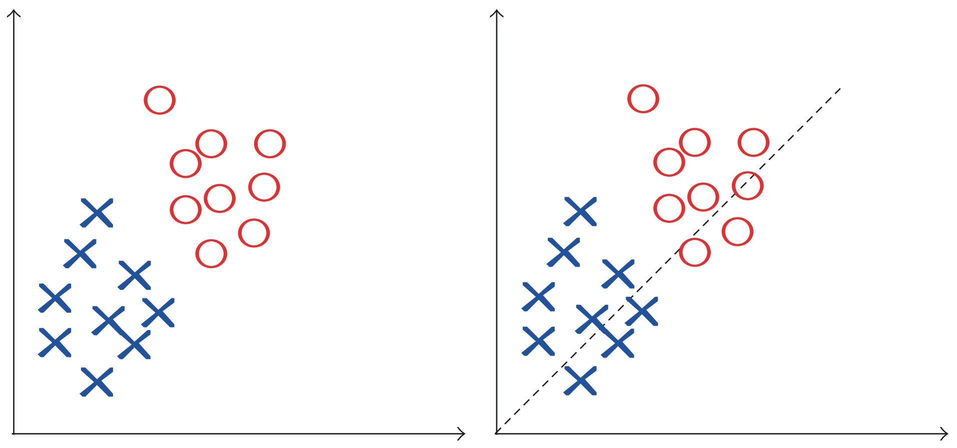
4.1. Fisher Classification. Fisher method actually is about the dimension compression. Projecting the samples which can be easily separated in higher space on a straight line arbitrarily may be difficult to be identified for different types mixed together. Generally, the best direction can always be found to separate the samples when projected on that direction. But how to find out the best direction and how to realize the transformations of projection toward the best direction are the very two problems to be solved by Fisher algorithm. Figure 4 shows analysis schematic diagram of Fisher algorithm using linear discriminant. In Figure 4(a), the sample cannot be identified when being projected on coordinate axis, and in Figure 4(b), the projection samples can be identified by looking for a direction.

4.2. Potential Function Classification. Potential function, a common method used in nonlinear classifier, is a way to solve the classification problems of pattern via the conception of electric field. In the potential function classification, the samples belonging to one category are treated as positive charge while the samples belonging to another category are treated as the negative charge, thus turning the classification problems of pattern to the matter of transferring the positive



(a) The subclass follows Gaussian distribution (b) The subclass does not follow Gaussian distribution

Figure 3: Judge whether each subclass follows Gaussian distribution.



(a) The sample cannot be identified when being projected on coordinate axis (b) The projection samples can be identified by looking for a direction

Figure 4: Schematic diagram of Fisher linear discriminant analysis.

charge and negative charge, and the equipotential line where its electric potential is zero is the decision boundary. The training course of potential function algorithm is a process of accumulating electric potential when the samples are input one after another by exploiting the potential function.

4.3. Least Squares Support Vector Machine (LS-SVM). Based on the VC dimension theory of statistical learning theory and the structural risk minimization principle, support vector machines method [12] converts the practical problem to high-dimensional feature space through nonlinear transformation and realizes the nonlinear discriminant function in the original space by constructing linear discriminant function in higher space. By means of introducing the least squares linear system into support vector machine to replace the traditional one, quadratic programming method, which is adopted to settle the problems of classification and estimation, is a kind of extension of traditional support vector machine.

5. Procedures of Electrofacies Analysis

The procedure of electrofacies analysis is shown in Figure 5.

5.1. Feature Extraction of Log Data. The primary step to establish electrofacies is to extract a set of log data features that can reflect the lithologic character of sedimentary rock. Generally, there are 9 types of well-logging items or more and those logging items are interrelated. There are two ways to eliminate gibberish, simplify control methods, and reduce calculated amount: (1) principal component analysis. (2) Select logging items manually. The extracted logging items will be recorded in Table stdlogdata as the data source for clustering analysis.

5.2. Clustering Analysis. In order to find out the electrofacies of the same type and establish a standard library in electrofacies analysis, clustering analysis must be conducted to stratum. Finally, the classification results acquired by clustering

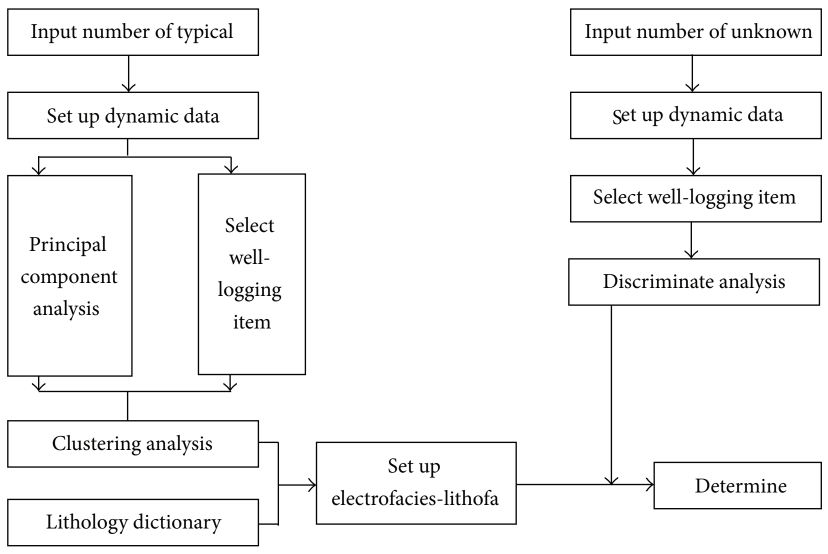


Figure 5: Flow diagram of electrofacies analysis.

should be recorded in the column of "Category" in Table stdlogdata and the lithology be recorded in the column of "Lithology" according to the lithology dictionary.

5.3. Discriminant Analysis. After establishing lithofacies database, namely, the electrofacies of type well, it is possible to discriminate the lithofacies of other wells. After discrimination, the data and discriminant results will be written in Table anylogdata and the logging items be written in the Table anylogitem.

6. Comparison and Analysis of Results of Algorithm

6.1. Experimental Data. The Iris dataset [13] usually serves as the testing dataset for benchmark function, in which each record contains 4 attributes of Iris, totaling 150 samples. The correct classification result is that each type of data has 50 samples. Eight attributes are included in each set of data of electrofacies, totaling 177 samples. As for the real data in electrofacies, there is no strictly accurate number of categories and standard classification. Judging by experience, 8 classifications may be rational.

6.2. Analysis of Experimental Results of Cluster

6.2.1. Iris Dataset. It can be easily seen from Figures 6–9 that the cluster obtained by standard K-means algorithm is pretty different from the standard results, while the clustering results obtained by ISODATA and KMRIC come near to the standard ones and are the same as the results obtained by built-in K-means algorithm of Matlab. AKM has only two categories. The second and the third categories are deemed as belonging to the same normal distribution that are never apart for they are approximate to each other and have some parts overlapped (see Table 1 and 2).

Table 1: Clustering method comparison under Iris dataset.

	K-means	ISODATA	KMRIC	AKM	Matlab
Type I	30	50	50	53	50
Type II	24	39	39	97	38
Type III	96	61	61	0	62
Accuracy	69.3%	92.6%	92.6%	66.7%	92%

Table 2: Clustering method comparisons under Iris dataset.

	K-means	ISODATA	KMRIC	AKM	Matlab
Type I	56	35	47	46	47
Type II	38	30	46	40	45
Type III	36	26	26	26	23
Type IV	18	23	14	23	23
Type V	11	17	13	13	13
Type VI	9	13	12	12	12
Type VII	8	12	10	10	11
Type VIII	1	10	9	7	3
Type IX	0	8	0	0	0
Type X	0	3	0	0	0

6.2.2. Electrofacies Dataset. It can be seen from Figures 10–13 that the clustering results obtained by K-means have large error, while the cluster obtained by KMRIC and AKM is relatively rational and can basically reflect the right classification, and AKM can also identify the accurate number of clustering automatically. Compared with ISODATA, AKM is more accurate in determining the number of clustering and its clustering results are more rational as well. Besides, it proves that the hypothesis testing way to judge the number of clustering of AKM is more universal than that by judging it based on the between-class distance of ISODATA.

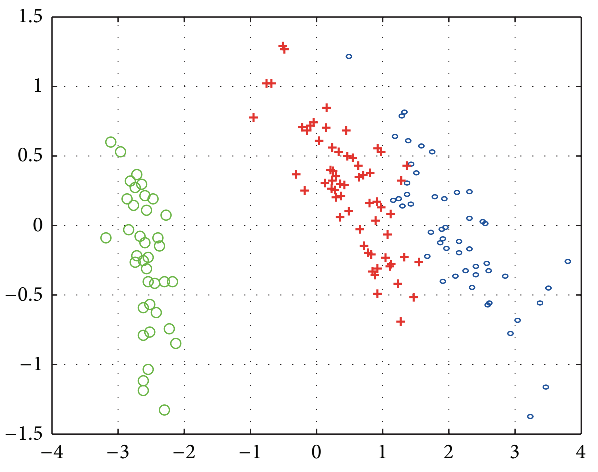


FIGURE 6: Clustering results of dataset by Matlab figure.

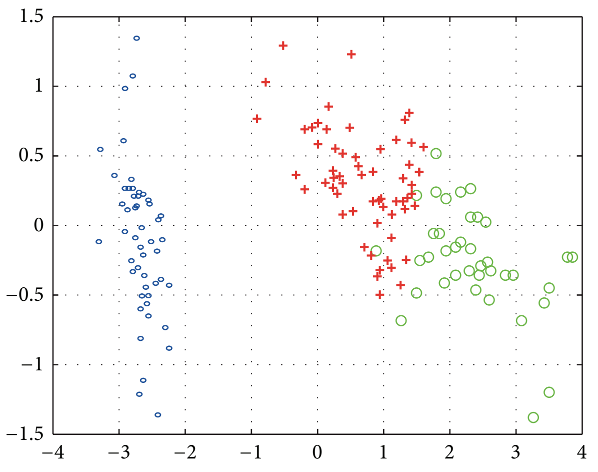


FIGURE 8: Clustering results obtained by ISODATA and KMRIC.

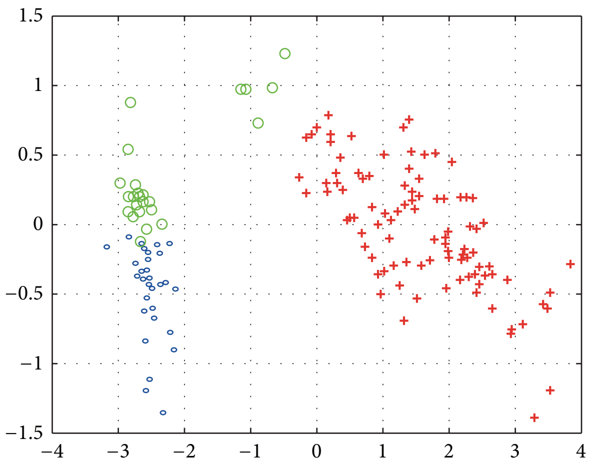


FIGURE 7: Clustering results obtained by standard K-means.

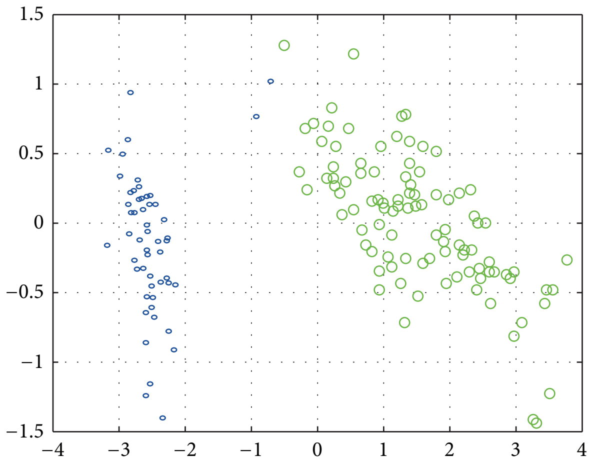


FIGURE 9: Clustering results obtained by AKM.

TABLE 3: Number of misclassification and accuracy of various discriminant methods under Iris dataset.

	Fisher	Potential function	LS-SVM
Type I	0	0	0
Type II	1	0	0
Type III	0	0	0
Total	1	0	0
Accuracy	96.7%	100%	100%

TABLE 4: Number of misclassification of various discriminant methods under electrofacies dataset.

	Fisher	Potential function	LS-SVM
Type I	—	0	0
Type II	—	0	2
Type III	—	0	0
Type IV	—	1	2
Type V	—	0	0
Type VI	—	0	0
Type VII	—	0	3
Type VIII	—	0	2
Total	—	1	9
Accuracy	—	94.9%	76.9%

6.3. Experimental Results and Analysis of Classification

6.3.1. Iris Dataset. See Table 3.

6.3.2. Electrofacies Dataset. It can be seen from Tables 3 and 4 that these three classification methods all work well when processing the Iris data for the data structure of Iris is quite simple and low in dimension. As for electrofacies data, Fisher discriminant analysis is not applicable due to the singular within-class scatter S_w matrix, while the potential function and LS-SVM still have better accuracy to classification. The multiclassification of LS-SVM application remains for further study.

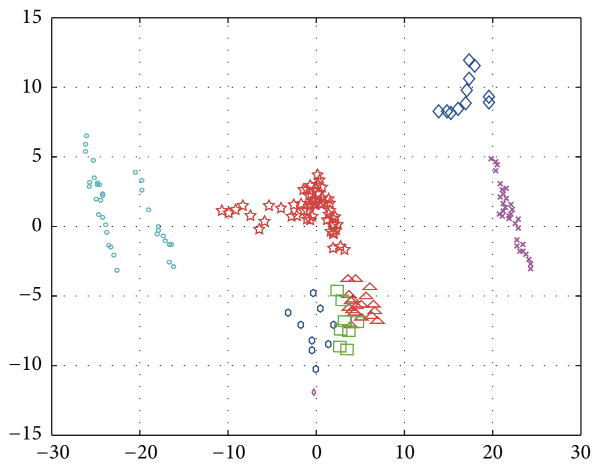


FIGURE 10: Clustering results obtained by standard K-means.

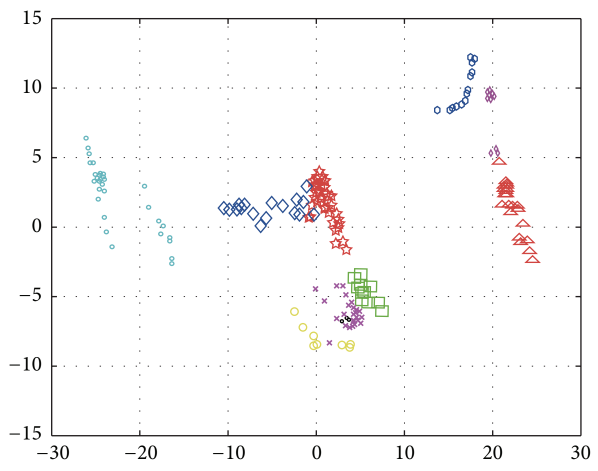


FIGURE 11: Clustering results obtained by ISODATA.

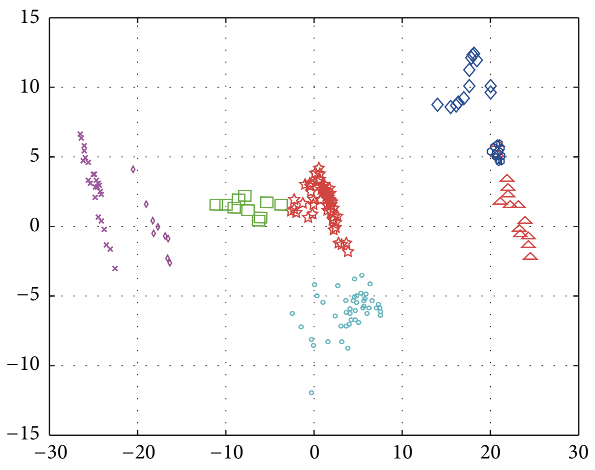


FIGURE 12: Clustering results obtained by KMRIC.

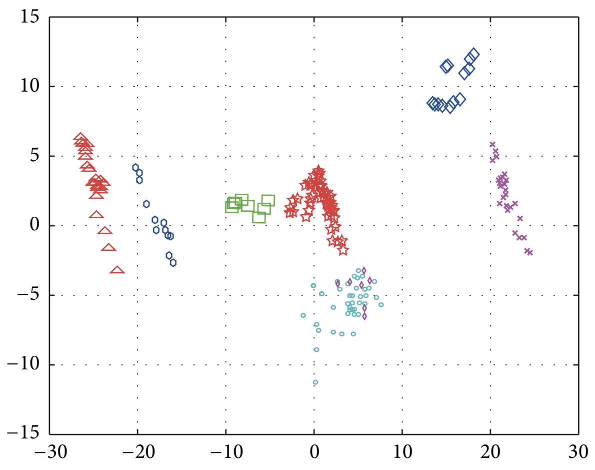


FIGURE 13: Clustering results obtained by AKM.

7. Conclusion

On the basis of analyzing the strengths and weaknesses of the existing main algorithms for clustering, this paper proposed the KMRIC algorithm for improving initial points and the AKM algorithm for determining the number of clusters. The support vector machine has also been used for classification. Finally, the algorithms are applied to electrofacies analysis. Through the experimental analysis, comparison was made among algorithms. According to the experimental results, the KMRIC algorithm erases the sensibility of K-means algorithm to data input sequence and initial centers, and it achieves an obvious improvement relative to K-means and ISODATA; AKM algorithm mixes the process of determining the number of clusters and the clustering process together to avoid the subjectivity in selecting the number of clusters and the blindness in initial divisions. Under general condition, the number of clusters and rational clusters can be found correctly.

There are some other problems that remain open. The volatility of results, which was caused by the randomness of selecting initial points in KMRIC, existed in KMRIC and AKM. To address this problem, we can lower the randomness by selecting the optimal initial points repeatedly. Hierarchical clustering is a very stable method but its disadvantage is the massive calculation cost. How to combine the hierarchical clustering and the abovementioned methods may be taken as the improvement direction in future.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

Research of the authors was supported by the National Nature Science Foundation of China (no. 61103037), Nature Science Foundation of Guangdong Province (no. S2013010011858), Project of Guangdong University of Outstanding Young Talents Cultivation (no. 2012LYM_0125), and Dongguan Science and Technology Project (no. 2012108102007).

References

[1] S. Dong, D. D. Zhou, and W. Ding, "Flow cluster algorithm based on improved K-means method," *IETE Journal of Research*, vol. 59, no. 4, pp. 326–333, 2013.

[2] J. Q. He, H. Dai, and X. Song, "The combination stretching function technique with simulated annealing algorithm for global optimization," *Optimization Methods and Software*, vol. 29, no. 3, pp. 629–645, 2014.

[3] J. Liu and T. Z. Liu, "Detecting community structure in complex networks using simulated annealing with k-means algorithms," *Physica A: Statistical Mechanics and Its Applications*, vol. 389, no. 11, pp. 2300–2309, 2010.

[4] S. H. Kim and L. Li, "Statistical identifiability and convergence evaluation for nonlinear pharmacokinetic models with particle swarm optimization," *Computer Methods and Programs in Biomedicine*, vol. 113, no. 2, pp. 413–432, 2014.

[5] S. Kalyani and K. S. Swarup, "Particle swarm optimization based K-means clustering approach for security assessment in power systems," *Expert Systems with Applications*, vol. 38, no. 9, pp. 10839–10846, 2011.

[6] D. H. Wang, J. F. Wang, and X. Y. Xu, "A relevance vector machine and bare-bones particle swarm optimization hybrid algorithm for PD pattern recognition of XLPE cable," *Journal of Computational Information Systems*, vol. 8, no. 2, pp. 451–458, 2012.

[7] D. Pelleg and A. W. Moore, "X-means: extending K-means with efficient estimation of the number of clusters," in *Proceedings of the 17th International Conference on Machine Learning*, pp. 727–734, 2000.

[8] E. Bertini, A. Tatu, and D. Keim, "Quality metrics in high-dimensional data visualization: an overview and systematization," *IEEE Transactions on Visualization and Computer Graphics*, vol. 17, no. 12, pp. 2203–2212, 2011.

[9] L. M. Li and Z. S. Wang, "Method of redundant features eliminating based on k-means clustering," *Applied Mechanics and Materials*, vol. 488, pp. 1023–1026, 2014.

[10] C. H. Lin, C. C. Chen, H. L. Lee et al., "Fast K-means algorithm based on a level histogram for image retrieval," *Expert Systems with Applications*, vol. 41, no. 7, pp. 3276–3283, 2014.

[11] P. A. W. Lewis, "Distribution of the Anderson-Darling statistic," *Annals of Mathematical Statistics*, vol. 32, pp. 1118–1124, 1961.

[12] M. Z. Tang and C. H. Yang, "Excellent operational pattern recognition based on simultaneously optimizing cost-sensitive support vector machine," *CIESC Journal*, vol. 64, no. 12, pp. 4509–4514, 2013.

[13] C. L. Blake and C. J. Merz, "UCI repository of machine learning databases," 1998.

Hindawi Publishing Corporation
Journal of Applied Mathematics
Volume 2014, Article ID 138370, 10 pages
http://dx.doi.org/10.1155/2014/138370

Research Article

Formal Modeling and Analysis of Fairness Characterization of E-Commerce Protocols

Chengwei Zhang, Xiaohong Li, Jing Hu, Zhiyong Feng, and Jiaojiao Song

School of Computer Science and Technology, Tianjin University, Tianjin 300072, China

Correspondence should be addressed to Xiaohong Li; xiaohongli@tju.edu.cn

Received 24 January 2014; Revised 4 May 2014; Accepted 4 May 2014; Published 1 June 2014

Academic Editor: Xiaoyu Song

In the past, fairness verification of exchanges between the traders in E-commerce was based on a common assumption, so-called nonrepudiation property, which says that if the parties involved can deny that they have received or sent some information, then the exchanging protocol is unfair. So, the nonrepudiation property is not a sufficient condition. In this paper, we formulate a new notion of fairness verification based on the strand space model and propose a method for fairness verification, which can potentially determine whether evidences have been forged in transactions. We first present an innovative formal approach not to depend on nonrepudiation, and then establish a relative trader model and extend the strand space model in accordance with traders' behaviors of E-commerce. We present a case study to demonstrate the effectiveness of our verification method.

1. Overview

The E-commerce protocol is a special kind of security protocol aiming to coordinate the exchange of valuable information between traders. The fairness of E-commerce protocols is the essential property and has become a research hotspot in recent years. E-commerce protocols are different from traditional cryptographic protocols, and common security analysis methods are invalid to fairness validating. It is common to extend existing methods like Kailar logic [1], SVO logic [2], CSP process algebra [3], and strand space model [4–6] to analyze such protocols. These methods are based on the assumption that the nonrepudiation has been established, and then verify fairness of fair exchange protocols. However, fairness validating of E-commerce protocol needs to verify not only the exchange process of protocols but also the evidences exchanged between traders in transactions.

To verify fairness of E-commerce protocols, we present a strand space model based fairness verification method in this paper, which is independent of nonrepudiation. The concept of fairness is decomposed into fair exchange of evidences and fair evidences of exchange, and a formal definition of fairness and fairness evidences is introduced. The trader model is constructed according to trader's behaviors, which is different from Dolev-Yao penetrator model [7]. The strands, in which no evidences from its opponent are obtained by the entity, are analyzed; thus, the nontermination dilemma caused by the state space explosion problem can be avoided.

The paper is organized as follows: related work on fairness verification is discussed in Section 2. Some background knowledge about strand space theory and the Dolev-Yao penetrator model is introduced in Section 3. The core body of the paper is followed in Section 4, which consists of definition and analysis of fairness as well as the discussion of trader models and the extended theories of the strand space model. The improved strand space model is tested by using the EMH protocol in Section 5, which proposes an improved protocol. Finally, the paper's conclusion is offered in Section 6.

2. Related Works

The E-commerce protocol has two objectives: the first one enables each protocol participant to seamlessly exchange valuable information. The second one enables each protocol participant to ensure the exchangeable fairness. The fair exchange protocol is a basic protocol of various E-commerce applications. From the perspective of protocol structure, it can be divided into three categories: gradual exchange protocols, on-line TTP exchange protocols, and offline TTP exchange protocols. In the 1980s, the appearance of gradual exchange protocols gradually increases the probability of correctness over several rounds of communication, but these protocols only can do progressive fairness [8]. The third party protocol requires a trusted third party. The third party is on-line which is required to be active in every exchange transaction [9]. These protocols relying heavily on TTP could easily lead to overload of networks and susceptible attacks. Offline fair exchange protocols have two phases: message exchange phase and dispute resolution phase. TTP was used only in the dispute resolution phase. This type of protocol reduces the problem of TTP as a source of bottleneck, because TTP is used very rarely and not involved in every round of exchange. Currently, most E-commerce protocols are based on offline TTP exchange protocols.

Formal analysis of security in E-commerce protocols is carried out more frequently than the proposal of traditional security protocols. For this reason, a variety of theories have been proposed, such as Kailar logic, SVO logic, and analysis method based on the CSP process algebra. Kailar proposed the concept "accountability," and a kind of logic to analyze accountability named Kailar logic. Accountability refers that protocol participants an prove that they have done something. Kailar logic verifies accountability by analyzing whether the parties have obtained the evidence that demonstrates the occurrence of the exchange. Zhou Jianying and Dieter Gollman put forward concepts EOO (nonrepudiation evidences of origin) and EOR (nonrepudiation evidences of receipt), which are used as the evidences of accountability. A nonrepudiation protocol allows two potentially mistrusting parities to exchange an electronic message together with EOO and EOR over the Internet in a fair way, that is, each party gets the other's term(s) or neither party does. They use SVO logic to verify nonrepudiation protocols. Steve Schneider uses CSP process algebra to analyze nonrepudiation protocols. The method he proposed can be utilized to analyze accountability and fairness. However, the analysis of the exchangeable fairness is based on on-line TTP rather than on offline TTP protocols which own a branched structure.

It is a prevailing approach for researchers to extend the strand space model for the analysis of E-commerce protocols in recent years. Yang and Deng [10] extended the strand space model to analyze TLS and IKE protocols. They proposed semiregular entities to denote entities which are different from penetrator and regular entities. Wang et al. [11] employed not only nonrepudiation EOO and EOR as the fairness evidences but also a similar method to prove the fairness of the iKP protocol as well. At the same time, Liu et al. [12] used a similar method to prove the fairness of the IBS protocol. These studies have provided a detailed description and analysis of E-commerce as well as its security properties.

All these methods verifying fairness are based on the prerequisite of guaranteeing exchangeable testimony as the satisfied nonrepudiation. They use nonrepudiation evidences as the fairness evidences. Nonrepudiation means that counterparts cannot deny that they have received or sent some information [13]. The E-commerce protocol requires not only fair exchange of evidences, but also the equivalence of exchange evidences.

The methods mentioned above are limited in the scope of the analyzed protocol, first of all, not all E-commerce protocols use TTP as an arbitration and fault-tolerant process [14]; in addition, the nonrepudiation evidences are the evidences of participants having sent or received information, while the fairness evidences are the valuable, quantifiable information exchanged by traders such as electronic money, bills, and signatures in running of an E-commerce protocol. These methods must assume that traders cannot forge evidences first and then analyze the fairness of exchange processes. Therefore, the study needs to explore a formal definition of fairness which does not rely on nonrepudiation.

In addition, Fröschle [15] put forward the concept branch of the strand space model. Branch describes the different choices of entities in an E-commerce protocol, which helps to traverse all the behaviors of protocol entities. Guttman [16, 17] defined fairness evidences as a collection of valuable information and then tracked all steps of a run to prove the fairness of the exchange protocol. Strand space model uses strands to describe all possible behaviors of protocol participants, and needs more than one strand to describe an entity's behaviors, making the model complex owing to the fact that participants in E-commerce protocols are possibly dishonest. At the same time, as the analysis of the fairness evidences requires considerable understanding of a protocol, so fairness can't be verified automatically. In addition, the space of entities' behaviors may be unlimited because of dishonesty, and the traversal may not be terminated; thus, a fairness verification method not to traverse all possible behaviors of protocol's entities needs to be explored.

Besides, although Guttman [16, 17] and we both extend the strand space model to verify fairness, the details of verification process between us are quite different. Guttman developed a model connecting protocol execution with state and state change and defined a new notation named state synchronization events to synchronize states between protocol participants. The "fair" in "fair exchange" in their definition refers to the balanced evolution of the state. In our method, we define the trader model to restrict traders' behaviors, and analyse all the possible results of a protocol. Their fairness focus on the transaction process, while we concentrate on the transaction result.

By the way, the protocol used as an instance to test our method in Section 5 has been proved to be unfair [18]. They analyzed and improved the EMH protocol, while the method they proposed is very purposeful and can only be used in few protocols [19].

3. Basic Concepts of Strand Space Model

Strand space model was proposed by Fabrega, Herzog, and Guttma in 1998, which analyzes security protocols in a hybrid analysis method combined to theorem proving and trace. It was proposed to formally analyze authentication and confidentiality of security protocols at first. There are some basic definitions of stand space model as follows.

A denotes the set of messages that can be exchanged between principals in a protocol. Terms are the elements of A. In a protocol, principals can either send or receive terms. To strand space model, the positive sign represents sending a term, whereas the negative sign represents receiving a term according to its occurrence.

Definition 1. A signed term is a pair $\langle \sigma, a \rangle$ with $a \in A$ and $\sigma \in \{+, -\}$. We will write a signed term as $+t$ or $-t$. $(\pm A)^*$ is the set of finite sequences of signed terms. We will denote a typical element of $(\pm A)^*$ by $\langle\langle \sigma_1, a_1 \rangle, \ldots, \langle \sigma_n, a_n \rangle\rangle$.

We extend the notion term to describe behaviors by traders in E-commerce protocols.

Definition 2. A strand space is a set Σ with a trace mapping $\mathrm{tr} : \Sigma \to (\pm A)^*$, where Σ is the set of strands.

Definition 3. A node is a pair $\langle s, i \rangle$, with $s \in \Sigma$ and i, an integer satisfying $1 \le i \le \mathrm{length}(\mathrm{tr}(s))$. The set of nodes is denoted by N. One will say the node $\langle s, i \rangle$ belongs to the strand s.

Definition 4. E is the set of edges. $n_1, n_2 \in N$ and $n_1 \to n_2 \in E$ means $\mathrm{term}(n_1) = +a$ and $\mathrm{term}(n_2) = -a$; $n_1 \Rightarrow n_2 \in E$ means n_1, n_2 occurs on the same strand with $\mathrm{index}(n_1) = \mathrm{index}(n_2) - 1$.

The actions available to penetrators are encoded in a set of penetrator traces that summarize the ability to discard messages, generate well-known messages, and piece messages together and apply cryptographic operations using keys that become available to him.

Definition 5. Penetrator model is defined by penetrator traces according to Dolve-Yao model assumptions:

M: text message: $\langle +t \rangle$, where $t \in T$,

F: flushing: $\langle -g \rangle$,

T: tee: $\langle -g, +g, +g \rangle$,

C: concatenation: $\langle -g, -h, +gh \rangle$,

S: separation into component: $\langle -gh, +g, +h \rangle$,

K: key: $\langle +K \rangle$, where $K \in K_P$, K_P is the set of keys initially known to the penetrator,

E: encryption: $\langle -K, -h, +\{h\}_k \rangle$,

D: decryption: $\langle -K^{-1}, -\{h\}_K, +h \rangle$.

Trace M means penetrator could send the messages they owned to the channel. While F means that they could obtain every message traveled in channel. T means repetition. C and S, respectively, represent joining and decomposition. E and D are the encryption and decryption. This set of penetrator traces ensures that the values that may be emitted by the penetrator are closed under joining, encryption, and the relevant "inverses" [5]. The trader model we proposed has the same form with penetrator model but models different abilities.

4. The Fairness Verification Using Strand Space Model

The paper focuses on E-commerce protocols containing third parties. The referred third party, here, are bank, arbiter, and trusted third party (TTP). Buyers and sellers in transactions exchange evidences, while third party guarantees fairness and effectiveness of the transactions. The model of third party is assumed to be regular and honest. This section is divided into three parts: the first part gives the formal definition of fairness and fairness evidences and then establishes the trader model and extends related concepts of the strand space model and, at last, gives a fair validation process based on those definitions.

4.1. The Formal Definition of Fairness. Fairness is one of the basic security properties that E-commerce protocols must meet; the acknowledged definition of fairness can be referred to [8–15]: an exchange is fair means that, at the end of the exchange, either each player receives the terms they expect or none receives any information about the other's terms. Fairness evidences are those terms that players expect in transaction. Fairness of E-commerce protocols includes the fair exchange of evidences and the fair evidences in exchange. The exchange in an electronic transaction is considered to be equivalent, so evidences exchanged in electronic transaction should be equivalent; otherwise, a trader may benefit from the other by forging unequal exchange evidences. To verify the fairness of the corresponding relation of evidences, the mapping relationship of defining evidences is in the following.

Definition 6. Traders $B_1, B_2 \in N$; N is the set of traders' identification; E_{B_1} and E_{B_2} are sets of fairness evidences belonging to traders B_1 and B_2, respectively. The evidence-corresponding-relation is a bijective function $f_{B_1} : E_{B_1} \to E_{B_2}$. Its inverse function is $f_{B_2} : E_{B_2} \to E_{B_1}$. $\forall e \in E_{B_1}$, $\exists! e' = f_{B_1}(e) \in E_{B_2}$, here, $f_{B_1}(e)$ means the evidences of B_2 which B_1 expects. The set of functions $f_{B_1}(e)$ is denoted by $F(E_{B_1}, E_{B_2})$. Similarly, the set from B_2 to B_1 is denoted by $F(E_{B_2}, E_{B_1})$.

Evidences exchanged in electronic transactions need to be mutually corresponded. Each trader has the ability to evaluate evidences, and the corresponding relations of traders' evidence act as an evaluation tool to measure equivalence of the evidence's value. We can verify whether traders have forged evidences from the formula $e' = f(e)$. If the formula is not established, it means that someone has forged evidences so that evidences exchanged are not equivalent. In our research, we assume that if a trader receives unequal evidence from others, they would like to ignore this message. This means that if a participant uses forged evidences in transaction, the other participant will refuse to accept the evidence.

The fair exchange of evidence, in the point of view of a trader, means that if A has not obtained the evidence from B, then B could not obtain A's evidence; else, if A obtained the evidence of B, whether B obtained evidence from A will not be considered anymore. Popular to say, the fairness means each participant of a transaction is not at a disadvantage. Combined with Definition 5, we define fairness as follows.

Definition 7. Given an E-commerce protocol Γ, here, $B_1, B_2 \in N$, $e_1 \in E_{B_1}$, $e_2 \in E_{B_2}$, E_{B_1}, and E_{B_2} are sets of evidences of B_1 and B_2. $f_1 \in F(E_{B_1}, E_{B_2})$ and $f_2 \in F(E_{B_2}, E_{B_1})$ are evidence-corresponding-relations between B_1 and B_2, respectively. Protocol Γ is fair, if and only if the following two conditions hold:

(1) at the end of an exchange, if trader B_1 has not obtained $f_1(e_1)$, then B_2 cannot get e_1;

(2) at the end of an exchange, if trader B_2 has not obtained $f_2(e_2)$, then B_1 cannot get e_2.

During the fairness validation process, we first generate a trader's strand which he has not got at the end of a transaction; the other trader acts as a penetrator to analyze whether the penetrator could obtain the evidence they want through a variety of deceptions or attacks from one of the strands above. The penetrator model here is different from the model used in the classical strand space model. Therefore, we need to model the behaviors of traders and extend related theories of the strand space model to describe E-commerce protocol and its properties.

4.2. Extension of Strand Space Model. Traders in E-commerce protocols are dishonest not like regular entities and penetrator entities in general security protocols. Apart from traders, all participants in E-commerce protocols are regular entities and perform in accordance with the agreement provisions. Because their sequence of events and entity model are constant, each of them has only limited numbers of strands. While traders may opt out of the transaction, or repeatedly use outdated evidences, like orders and electronic money, they may use a variety of behaviors to obtain benefits, thus the events sequence of these entities may not be complete in accordance with the protocol. Original strand space model uses the penetrator model to describe behaviors of attackers. A penetrator can intercept, send, forgery, tamper messages, and so forth. They can do almost everything except for resolving cryptographic algorithms, while traders are still bounded by the protocol. If using penetrator model to describe traders' behaviors, we might make a wrong judgment to a correct protocol. Thus, we need to establish a model to describe the behaviors of transactions between buyers and sellers. The capability of this model is between regular entities and penetrator entities. We call it trader model.

The study imitates the penetrator model in original strand space model to build the trader model and describe the behaviors of traders into atomic behaviors and their combinations. E-commerce protocols have little constrains for traders, similarly, traders cannot be completely separated from the protocol, so we use atomic behavior as well as some constraints on these atoms to describe the behaviors of traders. The model assumptions are showed as follows:

(1) traders can encrypt, decrypt, connect, and decompose message;

(2) traders can forge evidences and messages by using unequal evidences to get the other's interests;

(3) traders can send or receive messages belonging to this trader in protocol; only the messages are in specified format of protocol;

(4) both entities and messages in E-commerce protocols satisfy authentication and confidentiality, which being basic properties of security protocols; the protocol is insecure if these two properties are not met; fairness and nonrepudiation are based on the premise of security of the protocol; we should verify security before analysis fairness;

(5) the sequence of events in traders model needs not to be the order in accordance with the protocol; this assumption needs to describe the behaviors of traders in the form of atoms.

Considering E-commerce, protocols satisfy authentication and confidentiality; we add the identifiers of entities into the concept of the term, and terms in our model are represented as a 3-tuple. This modification eliminates the need for verification of authentication. Term is defined as follows.

Definition 8. Term is a 3-tuple $\langle \sigma, a, B \rangle$, where $\sigma \in \{+, -, \bigoplus\}$, $a \in A$, and $B \in N$. A is the message set and P is the set of entities' identifier. $\langle +, a, B \rangle$ means entity B sends message a; $\langle -, a, B \rangle$ means an entity receives message a from entity B; and $\langle \bigoplus, a, B \rangle$ means entity B owns message a. For convenience, abbreviate term $\langle \sigma, a, B \rangle$ as $\sigma a.B$; a is the unsigned portion of $\sigma a.B$.

We use symbol "$\bigoplus$" to meet the third assumption. Traders in E-commerce protocols cannot receive or send some messages, but can calculate. In the following, A_B is the set of terms that B could send or receive in protocol, and A_B^* is the set of messages B owns at the time.

Traders' atomic behaviors can be described by trace.

Definition 9. A trader trace is one of the following:

(1) M^*: text message: $\langle \bigoplus t.B, +t.B \rangle$, where $+t.B \in A_B$, $t \in A_B^*$, and $B \in N$; B is an entity's identifier; trace M^* expresses entity B sending message t;

(2) F^*: flushing: $\langle -t.C, \bigoplus t.B \rangle$, where $-t.C \in A_B$ and B, $C \in N$; trace F^* means entity B receives message t, and then B owns t;

(3) P^*: falsifying: $\langle \bigoplus e.B, \bigoplus f.B \rangle$, where $e \in E_B$ and $f \in F(E_B, E_{B'})$; E_B is the set of evidence belonging to entity B, $F(E_B, E_{B'})$ which is the set of evidence-corresponding-relations from B to B'; trace P^* expresses the situation that entity B forges evidence and its descriptions in protocol;

(4) C^*: concatenation: $\langle \bigoplus g.B, \bigoplus h.B, \bigoplus gh.B \rangle$, where $g, h \in A_B^*$ and $B \in N$; trace C^* means that if entity B owns messages g and h, then B owns gh;

(5) S^*: separation: $\langle \bigoplus gh.B, \bigoplus g.B, \bigoplus h.B \rangle$, where $gh \in A_B^*$ and $B \in N$; trace S^* means that if entity B owns message gh, then B owns g and h;

(6) K^*: key: $\langle \bigoplus K.B \rangle$, where $K \in K_B$, K_B is the set of keys B owns; trace K^* means entity B owns key K;

(7) E^*: encryption: $\langle \bigoplus K.B, \bigoplus h.B, \bigoplus \{h\}_K.B \rangle$, where $K \in K_B$, $h \in A_B^*$, and $B \in N$; trace E^*expresses the situation that entity B can encrypt h with key K;

(8) D^*: decryption: $\langle \bigoplus \{h\}_K.B, \bigoplus K^{-1}.B, \bigoplus h.B \rangle$, where $K \in K_B$, $\{h\}_K \in A_B^*$, and $B \in N$; trace D^* expresses the situation that entity B can decrypt $\{h\}_K$ into h if they own both $\{h\}_K$ and K^{-1}.

The strand of a trader is composed of nodes in traces M^* and F^*, expressing a sequence of events of the entity, and the remaining six traces are used to describe the entity's message space. Here, "+" and "−" denote messages delivered between entities, and "$\bigoplus$" stands for messages generated inside entities. The classical strand space model does not describe the concept "owned," because the number of messages sent or received by generators are unlimited. Messages which can be sent or received are owned by penetrators, while traders can only send or receive messages belonging to them. We propose the concept "owns" to describe the ability that a trader can process all the messages that they encountered. In summary, we modify the definition of terms to solve authentication and confidentiality, remove the tee trace T, add trace P^* and symbol "$\bigoplus$" to limit traders' abilities, and, finally, build the traders model.

Classical strand space model uses edges "$\rightarrow$" and "$\Rightarrow$" to describe the causal relationship between terms, where "$\rightarrow$" edge expresses delivering message between entities, and "$\Rightarrow$" edge describes an entity's state transition. Because traders' entities are semihonest, one may need more than one strand to describe their behaviors. Using atomic behaviors to build the trader model could describe the trader's behaviors nicely but is not conducive to express the causal relationship between nodes. For this purpose, we define owning set A_B^* and sending-receiving set A_B, where owning set A_B^* is a concept similar to ideal in the classical model and stands for messages that entity B has owned. The set is generated in a recursive way. Sending-receiving set A_B contains messages which B could send or receive and is fixed in a specific protocol. The owning set A_B^* is defined as follows.

Definition 10. A_B^* is an owning set of entity B; if $g \in A_B^*$, then

(1) $E_B \subseteq A_B^*$, $K_B \subseteq A_B^*$ and $F(E_B, E_{B'}) \subseteq A_B^*$, where E_B is the set of B's evidences, K_B is the set of B's keys, and $F(E_B, E_{B'})$ is the set of evidence-corresponding-relations from B to B';

(2) $\forall h \in A_B^*$, one has $gh \in A_B^*$;

(3) $\forall K \in K_B$, one has $\{g\}_K \in A_B^*$;

(4) If $g = g_1 g_2$, then $g_1, g_2 \in A_B^*$;

(5) If $g = \{g_1\}_K$ and $K^{-1} \in K_B$, then $g_1 \in A_B^*$.

Owning set A_B^* is used to describe all messages owned by entity B. The owning set of an entity will change with the state of the entity. When trader B receives messages S_B from their opponent, they could use those messages to obtain information from third party. If a strand of entity B includes flushing trace F^* : $\langle -h.C, \bigoplus h.B \rangle$, we have $h \in A_B^*$. We stipulate $A_B^*[h]$ as the owning set when entity B receives message h. If S_B is the set of messages entity B received from others at that time, then $A_B^*[S_B]$ will be all messages that entity B owns at the time they receive messages S_B.

Definition 11. Define owning set $A_B^*[S_B]$ as follows, where S_B is the set of messages received by entity B:

(1) $S_B \subseteq A_B^*[S_B]$, $E_B \subseteq A_B^*[S_B]$, $K_B \subseteq A_B^*[S_B]$, and $F(E_B, E_{B'}) \subseteq A_B^*[S_B]$, where E_B is the evidence set of entity B, K_B is the key set, and $F(E_B, E_{B'})$ is the set of evidence-corresponding-relations from B to B';

(2) $\forall g, h \in A_B^*[S_B]$, one has $gh \in A_B^*[S_B]$;

(3) $\forall g \in A_B^*[S_B]$, $\forall K \in K_B$, one has $\{g\}_K \in A_B^*[S_B]$;

(4) If $g = g_1 g_2 \in A_B^*[S_B]$, then $g_1, g_2 \in A_B^*[S_B]$;

(5) If $g = \{g_1\}_K \in A_B^*[S_B]$ and $K^{-1} \in K_B$, then $g_1 \in A_B^*[S_B]$;

(6) If $-a.B \in A_T \land +a.T \in A_B \land a \in A_B^*[S_B]$, in which there is a sequence $\langle -a.B, +c.\text{TTP} \rangle$ in T's strand, where $-c.\text{TTP} \in A_B$, then $c \in A_B^*[S_B]$.

The sixth rule expresses the situation in which trader B sends message a to third party T after they received messages S_B, and then T sends message c to B; thus entity B owns c. The third party here refers to all participators except traders in E-commerce protocol, including banks and arbitration institutions, as well as the trusted third party (TTP). Third party in E-commerce protocols is regular entity, and its strands are regular strands. For convenience, we use Definition 7 to define terms of regular strands. Edges between nodes remain unchanged, which are defined by "$\rightarrow$" and "$\Rightarrow$" in classical strands space model. In addition, we add a status node [15] at the end of each strand, which does not send or receive messages, only to be used to express the end of a sequence of events. In the description of strands, we follow the way Fröschle did in [15], using hollow circle and solid circle, respectively, to express the node of termination status and normal nodes.

4.3. Fairness Validation Process.

Based on the fairness definition and the extended strand space model above, we put forward a formal method of fairness verification. The verification process is shown in Figure 1.

Build $F(E_{B_1}, E_{B_2})$: each E-commerce protocol contains some description messages. These messages express the relationship of evidences which traders exchange. Therefore, we build evidence-corresponding-relations base on them. Because traders can forge evidences and description messages, we use set $F(E_B, E_{B'})$ to express all goods descriptions traders can forge.

Build T strand: all entities except traders in E-commerce protocol are regular entities; thus, we use a regular strand to model those entities. Each T has a fixed number of strands. We need to select T strands according to the implementation of protocols. For convenience, the study defines terms in

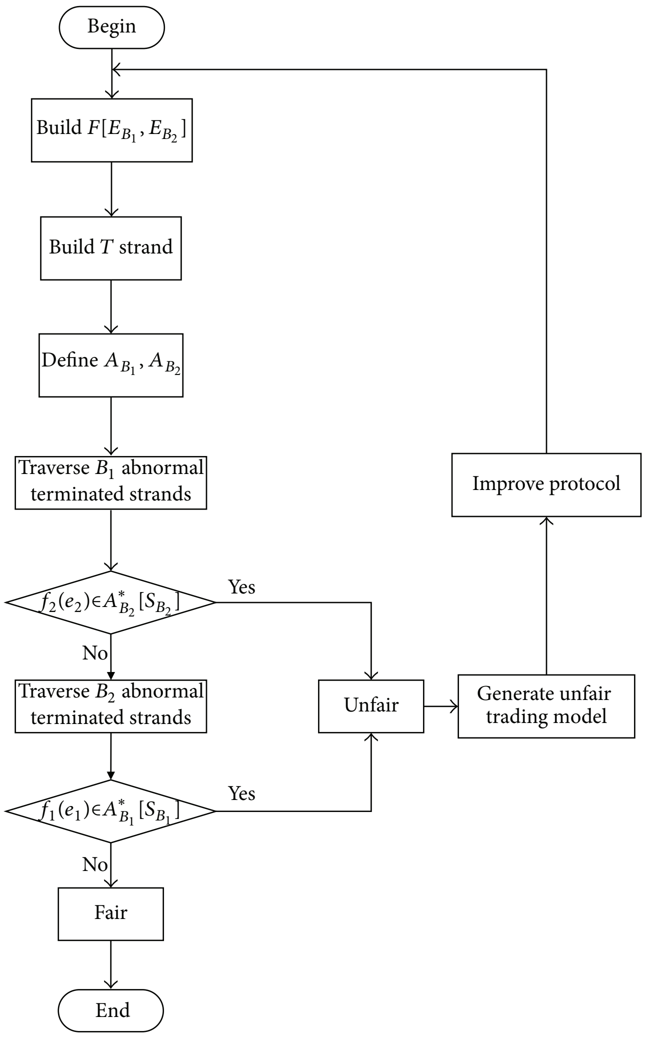

Begin
Build $F[E_{B_1}, E_{B_2}]$
Build T strand
Define A_{B_1}, A_{B_2}
Traverse B_1 abnormal terminated strands
$f_2(e_2) \in A^*_{B_2}[S_{B_2}]$
Yes
No
Traverse B_2 abnormal terminated strands
$f_1(e_1) \in A^*_{B_1}[S_{B_1}]$
Yes
No
Fair
End
Unfair
Generate unfair trading model
Improve protocol

Figure 1: Fairness verification schemes.

T strand by Definition 7 and adds a status node in the end of the strand.

Define A_{B_1} and A_{B_2}: sending-receiving sets A_{B_1} and A_{B_2} of traders B_1 and B_2 are fixed. Sending messages of a trader are all messages that protocol formatted, while receiving messages is not only protocol formatted but also meets the needs of their own interest.

Traverse traders' abnormal-terminated strands: abnormal terminated means a trader has not got the evidences they want at the end of a transaction. We first establish an abnormal-terminated strand of a trader with regular entities model and then detect whether the other trader could obtain the evidences they want. E-commerce protocols may be terminated abnormally in different running stages, and each trader may have more than one abnormal-terminated strand. We use regular strand to model trader's abnormal-terminated event sequence and detect whether a dishonest trader could gain evidences from an honest trader. Benefiting from assumptions, there will be a finite number of abnormal-terminated strands of each trader, which makes the detection process to be terminated possible.

Consider $f_1(e_1) \in A^*_{B_1}[S_{B_2}]$: $f_1(e_1)$ stands for the evidence B_1 expected; S_{B_2} stands for messages B_1 received from an abnormal-terminated strand of B_2; $A^*_{B_1}[S_{B_2}]$ stands for all messages B_1 owned after they receive messages S_{B_2}. $f_1(e_1) \in A^*_{B_1}[S_{B_2}]$ means that B_1 got the evidence they want, while B_2 did not; then, the protocol is unfair. The difficulty of this step is how to generate set $A^*_{B_1}[S_{B_2}]$. The recursively defined set $A^*_{B_1}[S_{B_2}]$ is an infinite set, and judgments about formula $f_1(e_1) \in A^*_{B_1}[S_{B_2}]$ need reasoning and induction. The specific process will be given in the next chapter.

Based on the strand space model, we use graphs to illustrate the process of an implementation of a protocol. If the protocol is unfair, an unfair execution process will be given by the strand space model in an intuitive way. For this reason, we modify the protocol and then verify the modified protocol again, until it is fair.

5. Case Analysis

We use the EMH protocol to test the fairness validation method which we propose. EMH protocol is an offline TTP electronic payment protocol proposed by Alaraj and Munro in 2007 [19]. The purpose of this protocol is to exchange a digital product (D) with a payment (P) between a customer (C) and a merchant (M). When we say that the protocol is fair, it means that, at the end of a transaction, either M gets P and C gets D or both of them do not get any message and vice versa. Using this protocol for the experiment can help to introduce the fairness verification process in detail and can explain the reason why we define fairness and extend the strand space model in such a way vividly.

5.1. Protocol Description. Identifier and symbol description includes the following:

C: customer,

M: merchant,

TTP: the trusted third party,

CB: the customer's bank, having the case that while the CB can also be considered as a TTP, TTP and CB are considered as third parties in our verification process and are modeled by regular entity,

D: digital product,

P: buyer's payment voucher, where D and P are the so-called fairness evidence in our model,

Desc: description of digital product, which is the link between D and P, where we build evidence corresponding relationship based on Desc, where $\mathrm{Desc}_C(P)$ and $\mathrm{Desc}_M(D)$ represent the evidence corresponding relationship between C and M, respectively,

$h(x)$: a strong collision-resistant one-way hash function, such as MD5,

PK_a: RSA public key of entity a,

SK_a: RSA private key of entity a,

P_cert: payment's certificate that is issued by the CB, with the contents of P_cert being d, description of payment (the amount), hp, hash value of payment, hep, hash value of encrypted payment with PK_a, and Sig_{CB}, CB's signature on P_cert,

Cert_{CT}: the certificate for the shared public key between C and TTP, which is issued by the TTP,

$\text{Enc}_{PK_a}(X) = \{X\}_{PK_a}$: an RSA encryption of X using the public key PK_a,

$\text{Dec}_{SK_a}(Y) = \{Y\}_{SK_a} = X$: an RSA decryption of Y using the private key SK_a,

$\text{Enc}_{PK_a}(X) = \{X\}_{PK_a}$: the RSA signature of party A, that is, encryption of the hash value of X using the private key SK_a,

$A \rightarrow B : X$: A which sends message X to B,

$X \| Y$: concatenation of messages X and Y.

EMH protocol is divided into three phases: the preexchange phase, the exchange phase, and the dispute settlement phase; details are given as follows.

(1) The preexchange phase includes the following:

mes1: $\text{TTP} \rightarrow C : \text{Cert}_{\text{CT}}$;

mes2: $\text{CB} \rightarrow C : P\_\text{cert}$.

(2) The exchange phase includes the following:

mes3: $C \rightarrow M : \text{Desc} \| \{P\}_{PK_{\text{CT}}} \| P\_\text{cert} \| \text{Cert}_{\text{CT}} \| \text{Sig}_C(P)$;

mes4: $M \rightarrow C : \{D\}_{PK_{\text{CT}}} \| \text{Sig}_M(D)$;

mes5: $C \rightarrow M : SK_{\text{CT}}$.

(3) The dispute settlement phase includes the following:

mes6: $M \rightarrow \text{TTP} : \text{Desc} \| \{P\}_{PK_{\text{CT}}} \| P\_\text{cert} \| \text{Cert}_{\text{CT}} \| \text{Sig}_C(P) \| \{D\}_{PK_{\text{CT}}} \| \text{Sig}_M(D)$;

mes7: $\text{TTP} \rightarrow C : \{D\}_{PK_{\text{CT}}} \| \text{Sig}_M(D)$;

mes8: $\text{TTP} \rightarrow M : SK_{\text{CT}}$.

The preexchange phase aims to award certificates from TTP and CB to C and do nothing between C and M, so we omit this phase in the verification process, considering only the last two stages.

5.2. Verification Process. To verify the fairness of EMH protocol, we need to prove that $f_C(P) \notin A_C^*[S_C]$ and $f_M(D) \notin A_M^*[S_M]$ both are true. First, we verify $f_C(P) \notin A_C^*[S_C]$; in the following proof, we could get $f_C(P) \in A_C^*[S_C]$.

5.2.1. Build the Set of Evidence Corresponding Relations. Define bijective functions $\text{Desc}_C : E_C \rightarrow E_M$ and $\text{Desc}_M : E_M \rightarrow E_C$, where $\text{Desc}_C \in F(E_C, E_M)$ and $\text{Desc}_M \in F(E_M, E_C)$. $F(E_C, E_M)$ and $F(E_M, E_C)$ are sets of evidence corresponding relations belonging to M and C, respectively. For $P \in E_C, D \in E_M$, and $\exists$ $\text{Desc} \in F(E_C, E_M)$ and its inverse $\text{Desc}^{-1} \in F(E_M, E_C)$, $D = \text{Desc}(P) \wedge P = \text{Desc}^{-1}(D)$.

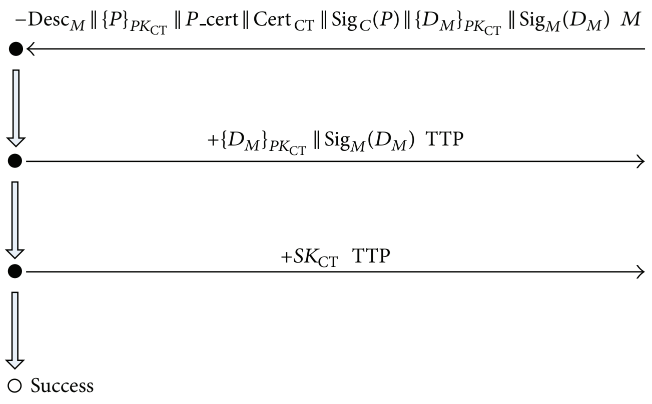

$-\text{Desc}_M \| \{P\}_{PK_{\text{CT}}} \| P\_\text{cert} \| \text{Cert}_{\text{CT}} \| \text{Sig}_C(P) \| \{D_M\}_{PK_{\text{CT}}} \| \text{Sig}_M(D_M)$ M
$+\{D_M\}_{PK_{\text{CT}}} \| \text{Sig}_M(D_M)$ TTP
$+SK_{\text{CT}}$ TTP
Success

FIGURE 2: Regular strand of TTP.

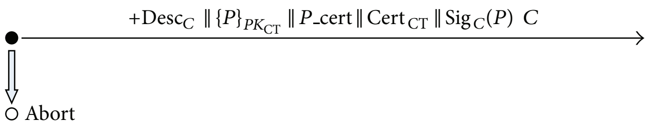

$+\text{Desc}_C \| \{P\}_{PK_{\text{CT}}} \| P\_\text{cert} \| \text{Cert}_{\text{CT}} \| \text{Sig}_C(P)$ C
Abort

FIGURE 3: An abnormal-terminated strand of trader C.

5.2.2. TTP Strands. TTP are regular entities, building its strands directly. Figure 2 is a regular strand of TTP.

TTP checks whether $D_M = \text{Desc}_M(P)$ is established; if so, it sends message $\{D_M\}_{PK_{\text{CT}}} \| \text{Sig}_M(D_M)$ to C and sends SK_{CT} to M.

5.2.3. Determine Sending-Receiving Set

Consider

$$\begin{aligned} A_C = \Big\{ & +\text{Desc}_C \Big\| \{P\}_{PK_{\text{CT}}} \| P\_\text{cert} \Big\| \text{Cert}_{\text{CT}} \Big\| \text{Sig}_C(P).C, \\ & -\{\text{Desc}_C(P)\}_{PK_{\text{CT}}} \Big\| \text{Sig}_M(\text{Desc}_C(P)).M, \\ & -\{\text{Desc}_C(P)\}_{PK_{\text{CT}}} \Big\| \text{Sig}_M(\text{Desc}_C(P)).\text{TTP}, \\ & +SK_{\text{CT}}.M \Big\} \\ A_M = \Big\{ & -\text{Desc}_C \Big\| \{P\}_{PK_{\text{CT}}} \| P\_\text{cert} \Big\| \text{Cert}_{\text{CT}} \Big\| \text{Sig}_C(P).C, \\ & +\{D_M\}_{PK_{\text{CT}}} \Big\| \text{Sig}_M(D_M).M, \\ & +\text{Desc}_M \Big\| \{P\}_{PK_{\text{CT}}} \| P\_\text{cert} \Big\| \text{Cert}_{\text{CT}} \Big\| \text{Sig}_C(P) \| \\ & \times \{D_M\}_{PK_{\text{CT}}} \Big\| \text{Sig}_M(D_M).M, \\ & -SK_{\text{CT}}.C, -SK_{\text{CT}}.\text{TTP} \Big\}. \end{aligned} \tag{1}$$

5.2.4. Establish Abnormal-Terminated Strands. We first analyze whether trader M could obtain the evidence P when trader C is abnormal-terminated strand (Figure 3).

5.2.5. Build Traders Model. We build M's trader model by the abnormal-terminated strand of C. Assuming that

M could obtain P; that is, there exists a flushing trace F^* : $\langle -SK_{\mathrm{CT}}.\mathrm{TTP}\rangle$ or $\langle -SK_{\mathrm{CT}}.C\rangle$. Because the abnormal-terminated strand of C does not contain node $\langle +SK_{\mathrm{CT}}.\mathrm{TTP}\rangle$, we ignore the situation F^* : $\langle -SK_{\mathrm{CT}}.C\rangle$.

Suppose the situation where there exists a node $\langle -SK_{\mathrm{CT}}.\mathrm{TTP}\rangle$ in M's strand. Because TTP strand is regular, there exists a node $\langle +\mathrm{Desc}\|\{P\}_{PK_{\mathrm{CT}}}\|P\_\mathrm{cert}\|\mathrm{Cert}_{\mathrm{CT}}\|\mathrm{Sig}_C(P)\|\{D_M\}_{PK_{\mathrm{CT}}}\|\mathrm{Sig}_M(D_M).M\rangle$ in M's strand.

And because $-\mathrm{Desc}\|\{P\}_{PK_{\mathrm{CT}}}\|P\_\mathrm{cert}\|\mathrm{Cert}_{\mathrm{CT}}\|\mathrm{Sig}_C(P).C \in A_M$, we have $S_M = \{\mathrm{Desc}\|\{P\}_{PK_{\mathrm{CT}}}\|P\_\mathrm{cert}\|\mathrm{Cert}_{\mathrm{CT}}\|\mathrm{Sig}_C(P).C\}$. Then, we analyze M's owning set $A_M^*[S_M]$.

By Definition 11, we have

$$\{P\}_{PK_{\mathrm{CT}}}\|P\_\mathrm{cert}\,\|\mathrm{Cert}_{\mathrm{CT}}\,\|\mathrm{Sig}_C(P) \in A_M^*[S_M] \quad (2)$$

$$D_M = \mathrm{Desc}_M(P),\mathrm{Desc}_M, D_M \in A_M^*[S_M] \quad (3)$$

$$\{D_M\}_{PK_{\mathrm{CT}}}, \mathrm{Sig}_M(D_M) \in A_M^*[S_M] \quad (4)$$

$$\mathrm{Desc}_M\left\|\{P\}_{PK_{\mathrm{CT}}}\right\|P\_\mathrm{cert}\,\|\mathrm{Cert}_{\mathrm{CT}}\,\|\mathrm{Sig}_C(P)\left\|\{D_M\}_{PK_{\mathrm{CT}}}\right\|\mathrm{Sig}_M(D_M) \in A_M^*[S_M]. \quad (5)$$

Because $\mathrm{Desc}_M\|\{P\}_{PK_{\mathrm{CT}}}\|P\_\mathrm{cert}\|\mathrm{Cert}_{\mathrm{CT}}\|\mathrm{Sig}_C(P)\|\{D_M\}_{PK_{\mathrm{CT}}}\|\mathrm{Sig}_M(D_M).M \in A_M$; there exists a text message trace M^* in M's strand, where M^* is $\langle +\mathrm{Desc}\|\{P\}_{PK_{\mathrm{CT}}}\|P\_\mathrm{cert}\|\mathrm{Cert}_{\mathrm{CT}}\|\mathrm{Sig}_C(P)\|\{D_M\}_{PK_{\mathrm{CT}}}\|\mathrm{Sig}_M(D_M).M\rangle$, which means that the assumption holds; trader M can obtain the evidence P. The protocol is unfair. Figure 4 is the unfair execution process described by strand space model.

5.2.6. Unfair Analysis and Protocol Improvements. Protocol is unfair because the TTP cannot accurately determine whether M has forged evidence. Trader M can obtain SK_{CT} from TTP by forging evidence D and the description message Desc and then get C's evidence. We modify protocol; thus the TTP could compare description information of the two parties and do a fair judgment.

Here is the modified protocol.

(1) The preexchange phase includes the following:

mes1: TTP $\rightarrow C$: $\mathrm{Cert}_{\mathrm{CT}}$;

mes2: CB $\rightarrow C : P\_\mathrm{cert}$.

(2) The exchange phase includes the following:

mes3: $C \rightarrow M$: $\mathrm{Desc}\|\{P\|\mathrm{Desc}\}_{PK_{\mathrm{CT}}}\|P\_\mathrm{cert}\|\mathrm{Cert}_{\mathrm{CT}}\|\mathrm{Sig}_C(P\|\mathrm{Desc})$;

mes4: $M \rightarrow C : \{D\}_{PK_{\mathrm{CT}}}\|\mathrm{Sig}_M(D)$;

mes5: $C \rightarrow M : SK_{\mathrm{CT}}$.

(3) The dispute settlement phase includes the following:

mes6: $M \rightarrow$ TTP : $\mathrm{Desc}\|\{P\|\mathrm{Desc}\}_{PK_{\mathrm{CT}}}\|P_{\mathrm{cert}}\|\mathrm{Cert}_{\mathrm{CT}}\|\mathrm{Sig}_C(P\|\mathrm{Desc})\,\|\{D\}_{PK_{\mathrm{CT}}}\|\mathrm{Sig}_M(D)$;

mes7: TTP $\rightarrow C : \{D\}_{PK_{\mathrm{CT}}}\|\mathrm{Sig}_M(D)$;

mes8: TTP $\rightarrow M : SK_{\mathrm{CT}}$.

Based on the unfair reasons above, we improve message 3 and message 6. TTP will determine $\mathrm{Desc}_C = \mathrm{Desc}_M \wedge D_M = \mathrm{Desc}_M(P)$ first, when they receive a request message from M. If the formula is true, TTP sends SK_{CT} to M and D_M to C.

By verifying the fairness of the modified protocol continuously, we then establish abnormal-terminated strands of M and C, respectively, and judge each of them. The concreted analysis process of improved protocol will not be described here. Figure 5 is the model description of the modified protocol in the same situation. From it, we can know that M has to send their evidence D in order to obtain P, because the TTP has more discrimination capability.

The verification steps are the same as mentioned above, so we did not propose the detailed description here.

5.3. Analysis of Experimental Results. By using EMH protocol to test the fair authentication method proposed in this paper, we draw some conclusions: first, the method can verify the fairness of E-commerce protocols effectively and give an accurate judgment about the fair exchange of evidences and the fair evidences in exchange; second, the method generates a finite number of abnormal-terminated strands and builds the trader model explicitly by inductive reasoning, which enables the verification process to be terminated; in addition, it has a practical value in design and improvement of E-commerce protocols.

6. Conclusion

This paper proposes a formal definition of fairness as well as a new method to verify the fairness of E-commerce protocols. The trader model we build here differs from the Dolev-Yao penetrator model. Because it is established according to the E-commerce trading behaviors, it can be better to reflect the behaviors of entities in E-commerce protocols. The evidence-corresponding-relations defined by a bijective function can describe the equivalent relations of traders' evidences and give a method to determine whether someone has forged evidences in transaction. The formal definition of fairness is defined from the perspective of traders, which helps to

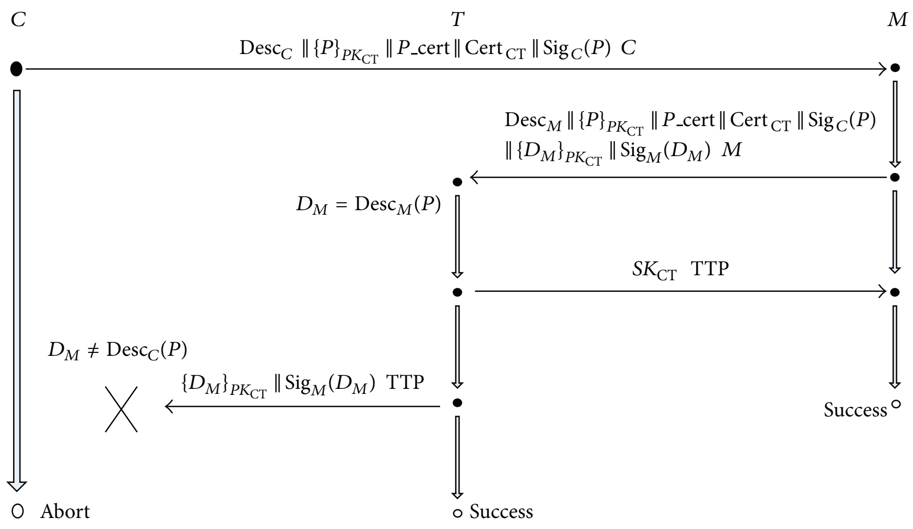

C
T
M
Desc_C ‖ {P}_PK_CT ‖ P_cert ‖ Cert_CT ‖ Sig_C(P) C
Desc_M ‖ {P}_PK_CT ‖ P_cert ‖ Cert_CT ‖ Sig_C(P) ‖ {D_M}_PK_CT ‖ Sig_M(D_M) M
D_M = Desc_M(P)
SK_CT TTP
D_M ≠ Desc_C(P)
{D_M}_PK_CT ‖ Sig_M(D_M) TTP
Success
Abort
Success

Figure 4: An unfair case of the protocol.

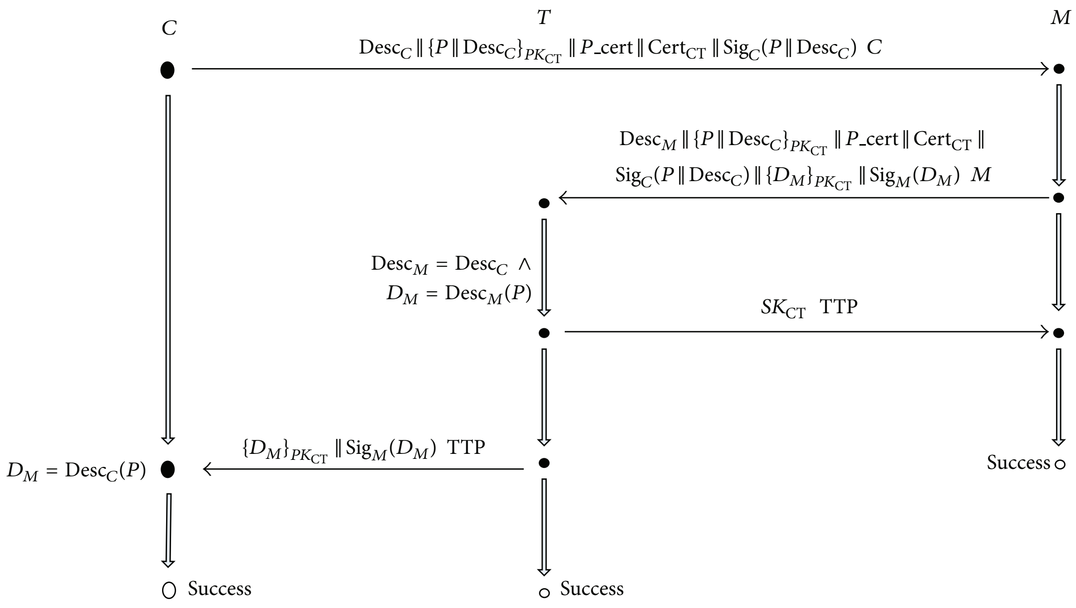

C
T
M
Desc_C ‖ {P ‖ Desc_C}_PK_CT ‖ P_cert ‖ Cert_CT ‖ Sig_C(P ‖ Desc_C) C
Desc_M ‖ {P ‖ Desc_C}_PK_CT ‖ P_cert ‖ Cert_CT ‖ Sig_C(P ‖ Desc_C) ‖ {D_M}_PK_CT ‖ Sig_M(D_M) M
Desc_M = Desc_C ∧ D_M = Desc_M(P)
SK_CT TTP
{D_M}_PK_CT ‖ Sig_M(D_M) TTP
D_M = Desc_C(P)
Success
Success
Success

Figure 5: Analysis of the modified protocol.

reconcile with model assumptions of traders. We use a regular strand to model the third party and trader abnormal-terminated strands, propose the trader model to detect whether a participant can obtain regular entities' evidences, and thus complete the fairness validation. This method avoids the verification of nonrepudiation, and can verify fairness of E-commerce protocols including third-parties. Besides, it neither needs to track all statuses of protocol execution nor traverses all strands of traders. The current work is limited to manual derivation, and we will strive to the automatic verification in future.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

This work was supported in part by the National Science Foundation of China (nos. 91118003, 61272106, and 61003080) and by 985 funds of Tianjin University.

References

[1] R. Kailar, "Accountability in electronic commerce protocols," *IEEE Transactions on Software Engineering*, vol. 22, no. 5, pp. 313–328, 1996.

[2] J. Zhou and D. Gollmann, "Towards verification of non-repudiation protocols," in *Proceedings of the International Refinement Workshop and Formal Methods Pacific*, pp. 370–380, Canberra, Australia, 1998.

[3] S. Schneider, "Formal analysis of a non-repudiation protocol," in *Proceedings of the 11th IEEE Computer Security Foundations Workshop (CSFW '98)*, pp. 54–65, June 1998.

[4] F. J. T. Fabrega, J. C. Herzog, and J. D. Guttman, "Honest ideals on strand spaces," in *Proceedings of the 11th IEEE Computer Security Foundations Workshop (CSFW '98)*, pp. 66–77, June 1998.

[5] F. J. T. Fabrega, J. C. Herzog, and J. D. Guttman, "Strand spaces: why is a security protocol correct?" in *Proceedings of the IEEE Symposium on Security and Privacy*, pp. 160–171, May 1998.

[6] F. J. T. Fabrega, J. C. Herzog, and J. D. Guttman, "Mixed strand spaces," in *Proceedings of the 12th IEEE Computer Security Foundations Workshop (CSFW '99)*, pp. 72–82, June 1999.

[7] R. M. Amadio and W. Charatonik, "On name generation and set-based analysis in the Dolev-Yao model," in *CONCUR 2002—Concurrency Theory*, pp. 499–514, Springer, Berlin, Germany, 2002.

[8] H. Pagnia and F. C. Gartner, On the Impossibility of Fair Exchange Without a Trusted Third Party TUD-BS-1999-02, Department of Computer Science, Darmstadt University of Technology, 1999.

[9] N. Asokan, *Fairness in electronic commerce [Ph.D. thesis]*, University of Waterloo, 1998.

[10] J. Yang and H.-F. Deng, "Security electronic commerce protocol by the third kind entities," in *Proceedings of the International Conference on Machine Learning and Cybernetics*, pp. 4438–4443, Dalian, China, August 2006.

[11] H. Wang, J. Ma, and B. Chen, "Formal analysis of fairness in E-payment protocol based on strand space," in *Web Information Systems and Mining*, pp. 469–478, Springer, Berlin, Germany, 2009.

[12] W. Liu, J. Yang, and Z. Li, "Fairness analysis of electronic commerce protocol based on strand space," in *Proceedings of the 5th International Conference on Intelligent Information Hiding and Multimedia Signal Processing (IIH-MSP '09)*, pp. 714–717, Kyoto, Japan, September 2009.

[13] W. Xu, D.-Y. Wu, Y. Ma, and N. Liu, "A formal method for analyzing fair exchange protocols," in *Proceedings of the WASE International Conference on Information Engineering (ICIE '09)*, pp. 117–120, Taiyuan, China, July 2009.

[14] Q. Zhang, K. Markantonakis, and K. Mayes, "A practical fair-exchange e-payment protocol for anonymous purchase and physical delivery," in *Proceedings of the IEEE International Conference on Computer Systems and Applications (AICCSA '06)*, pp. 851–858, Sharjah, UAE, March 2006.

[15] S. Fröschle, "Adding Branching to the Strand Space Model," *Electronic Notes in Theoretical Computer Science*, vol. 242, no. 1, pp. 139–159, 2009.

[16] J. D. Guttman, "State and progress in strand spaces: proving fair exchange," *Journal of Automated Reasoning*, vol. 48, no. 2, pp. 159–195, 2012.

[17] J. D. Guttman, "Fair exchange in strand spaces," in *Proceedings 7th International Workshop on Security Issues in Concurrency (SecCo '09)*, pp. 46–60, Bologna, Italy, September 2009.

[18] S.-H. Tian, L.-J. Chen, and J.-R. Li, "Fairness analysis of electronic payment protocol based on offline TTP," *Journal of Computer Applications*, vol. 29, no. 7, pp. 1839–1843, 2009.

[19] A. Alaraj and M. Munro, "An efficient fair exchange protocol that enforces the merchant to be honest," in *Proceedings of the 3rd International Conference on Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom '07)*, pp. 196–202, New York, NY, USA, November 2007.

Hindawi Publishing Corporation
Journal of Applied Mathematics
Volume 2014, Article ID 932397, 9 pages
http://dx.doi.org/10.1155/2014/932397

Research Article

A Software Vulnerability Rating Approach Based on the Vulnerability Database

Jian Luo, Kueiming Lo, and Haoran Qu

School of Software, Tsinghua University, Beijing 100084, China

Correspondence should be addressed to Jian Luo; j-luo10@mails.tsinghua.edu.cn

Received 14 March 2014; Accepted 14 May 2014; Published 29 May 2014

Academic Editor: Xiaoyu Song

CVSS is a specification for measuring the relative severity of software vulnerabilities. The performance values of the CVSS given by CVSS-SIG cannot describe the reasons for the software vulnerabilities. This approach fails to distinguish between software vulnerabilities that have the same score but different levels of severity. In this paper, a software vulnerability rating approach (SVRA) is proposed. The vulnerability database is used by SVRA to analyze the frequencies of CVSS's metrics at different times. Then, the equations for both exploitability and impact subscores are given in terms of these frequencies. SVRA performs a weighted average of these two subscores to create an SVRA score. The score of a vulnerability is dynamically calculated at different times using the vulnerability database. Experiments were performed to validate the efficiency of the SVRA.

1. Introduction

The common vulnerability scoring system (CVSS), developed and maintained by the CVSS special interest group (CVSS-SIG) working under the auspices of the forum for incident response and security teams (FIRST), can be applied to the classification of security vulnerability [1] and the analysis of attack models [2]. CVSS has been adopted by many software vendors and service providers [3]. The US federal government uses it for its National Vulnerability Database [4] and mandates its use in products validated by the security content automation protocol (SCAP) program.

There exist many proprietary schemes for rating software flaw vulnerabilities, most are created by software vendors, but CVSS is the only known open specification. In contrast to other scoring systems, CVSS was designed to be quantitative so that analysts would not have to perform qualitative evaluations of vulnerability severity. Great effort has been directed at developing the specification for CVSS so that any two vulnerability analysts should obtain identical CVSS scores for the same vulnerability. The scores are based on a series of measurements (called metrics) based on expert assessment.

1.1. Overview of CVSS Framework. CVSS provides an open framework for describing the characteristics and impacts of IT vulnerabilities. It contains three groups of metrics (see Figure 1), as explained in [5, 6].

(1) *Base.* It represents the intrinsic and fundamental characteristics of a vulnerability that are time-constant across user environments. An equation is applied to the values of the base metrics to compute a vulnerability's base score.

(2) *Temporal.* It represents the characteristics of a vulnerability that change over time but apply to all instances of a vulnerability in all environments, such as the public availability of an exploit code or a remediation technique. A temporal score for a vulnerability is calculated with an equation that uses both the base score and temporal metric values as parameters.

(3) *Environmental.* It captures the characteristics of a vulnerability that are associated with users IT environment. Since environmental metrics are optional, they each include a metric value that has no effect on the score. An environmental score is calculated with

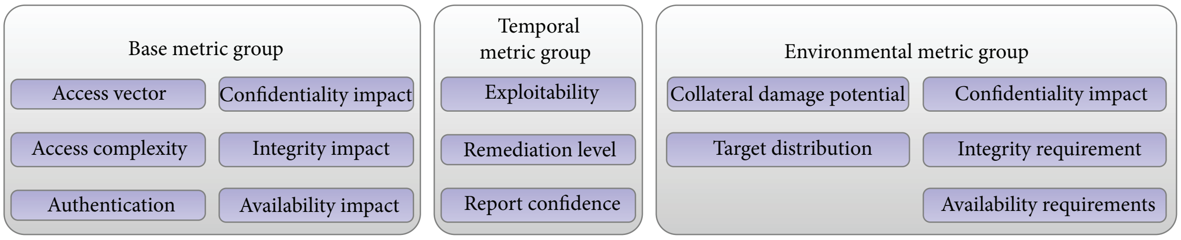


FIGURE 1: CVSS metric groups.

an equation that uses both the temporal score and the environmental metric values as parameters.

The initial CVSS specification was developed by the National Infrastructure Advisory Council and published in October 2004 [7]. During the analysis and use of the original CVSS version, many deficiencies were found, as explained in [8]. Finalized in 2007, the current version (CVSS v2) was designed to address these deficiencies. The base metric group of CVSS v2 has two subscores.

(1) *Exploitability subscore E*, composed of the access vector (AV), access complexity (AC), and authentication instances (AU), is computed by the following equation:

$$E = 20 \times \mathrm{AV} \times \mathrm{AC} \times \mathrm{AU}. \tag{1}$$

(2) *Impact subscore I*, which expresses the potential damage on confidentiality (CI), integrity (II), and availability (AI), is computed as follows:

$$I = 10.41 \times (1 - (1 - \mathrm{CI}) \times (1 - \mathrm{II}) \times (1 - \mathrm{AI})). \tag{2}$$

Table 1 gives all possible values of the six base metrics in v2, which are used to calculate these two subscores. The overall base score of v2 is expressed in terms of impact (I) and exploitability (E) components by

$$B = \begin{cases} 1.176 \times \left(\dfrac{3I}{5} + \dfrac{2E}{5} - \dfrac{3}{2}\right) & \text{if } I \neq 0, \\ 0 & \text{if } I = 0. \end{cases} \tag{3}$$

The base score is rounded to one decimal place and ranges from 0.0 to 10.0. More details related to CVSS metrics and their scoring computation can be found in the CVSS guide [5].

1.2. Shortcomings of CVSS v2. We downloaded 54,432 vulnerabilities listed in the common vulnerabilities and exposures (CVE) dictionary [9]; this encompasses all valid CVE entries published between 2002 and 2012. The scoring was performed by the national vulnerability database (NVD) [4] in accordance with the v2 specification.

When scoring separates vulnerabilities, they should be scored completely independently of each other and not take into account any interaction. According to v2 scoring tip number 1:

"Vulnerability scoring should not take into account any interaction with other vulnerabilities. That is, each vulnerability should be scored independently."

SPSS, a type of statistical software, was used to perform the Chi-square analysis, and the results indicate *there exist correlations among the six metrics*. For example, Table 2 shows the statistical data related to the frequencies of AV and AC. These data were used as inputs by SPSS, and the results of the Chi-square analysis are given in Table 3. Asymp. Sig. is smaller than the significance level 0.05, indicating that the correlation between AV and AC is significant. Similarly, there exist significant correlations between other metrics, for example, II and AI.

Most importantly, CVSS v2 fails to distinguish different vulnerabilities. As an example, a path disclosure flaw in a web application would be scored as (AV = Network, AU = None, AC = Low, CI = Partial, II = None, AI = None) for a total score of 5.0. A vulnerability that allows an attacker to traverse the file system and read any file accessible by the web server would receive the same score as the path disclosure flaw. These two flaws obviously pose significantly different risks, yet according to CVSS v2 standards, they are identical.

Once the v2 metrics were defined, opinions related to the scoring for each type of vulnerability were collected from the CVSS-SIG members and their organizations. Each of the six metrics had three possible values, resulting in 729 possible vulnerability types. It was not possible to create scores for these 729 types in the range [0.0, 10.0] in a justifiable manner. So, the researchers divided the base metrics into two subgroups: impact and exploitability. Each group had three metrics with three possible values, so only 27 vulnerability types per group had to be scored and ranked. The researchers reached consensus on the approximated rankings and scorings, leading to the creation of lookup tables for impact and exploitability. The CVSS score was computed by a weighted average of exploitability and impact. However, the CVSS community desired an equation instead of lookup tables. So, mathematicians proposed equations (1)–(3) to approximate the lookup tables. In essence, these equations were derived from the designers' experience and statistical results of vulnerability data. As time went on, it became clear that these equations, as well as the empirical values listed in Table 1, might no longer be applicable.

1.3. Our Design Methodology. To overcome the shortcomings mentioned above, a software vulnerability rating approach (SVRA) is proposed. SVRA takes time as an important parameter. Based on a vulnerability database, it counts the frequencies of the six metrics at any given time point. Then, the three values of each metric are given by their frequencies. As the frequencies change over time, each metric takes different values instead of a constant value.

The process of exploiting a vulnerability is a step-by-step procedure, but the impact is an evolutionary and accumulative process. So, the frequency of the vector (AV, AC, AU) is used to approximate the exploitability, while the frequencies of CI, II, and AI are utilized to calculate the impact. To create an SVRA score from these two subscores, SVRA also performs a weighted average of exploitability and impact, with exploitability having a weight of 0.4 and impact having a weight of 0.6. In terms of design methodology, SVRA is fundamentally different from CVSS v1 and CVSS v2. The score of a vulnerability in SVRA dynamically changes over time, which is not true in v2 or v1.

The rest of this paper is organized as follows. The next section provides the framework of the SVRA. Section 3 describes the analysis of and comparison between CVSS and SVRA, and the experimental results are also reported. Section 4 summarizes our conclusions and highlights some suggestions for future work.

2. Software Vulnerability Rating Approach

This section provides the framework of our software vulnerability rating approach (SVRA), where the base score of a vulnerability is dynamically calculated over time.

2.1. Frequency. Let $T \subseteq \mathbb{R}^+$ be the time domain. Obviously, all vulnerabilities have their own report times. Given $t \in T$, let $\mathscr{V}(t)$, a vulnerability database, be all vulnerabilities whose report time is less than or equal to t, and

$$\Omega = \{(\text{AV}, \text{Local}), (\text{AV}, \text{Adj.Net}), (\text{AV}, \text{Network}), \ldots, (\text{AI}, \text{None}), (\text{AI}, \text{Partial}), (\text{AI}, \text{Complete})\}, \tag{4}$$

which contains 18 elements. Then, for $\forall (m, a) \in \Omega$, a subset $\mathscr{V}(m = a, t)$ is defined as

$$\mathscr{V}(m = a, t) = \{v \in \mathscr{V}(t) \mid v.m = a\}. \tag{5}$$

The frequency of $m = a$ at t is denoted as $f(m = a, t)$ and it can be computed as follows:

$$f(m = a, t) = \begin{cases} \dfrac{\text{card}(\mathscr{V}(m = a, t))}{\text{card}(\mathscr{V}(t))} & \text{if } \mathscr{V}(t) \neq \emptyset, \\ 0 & \text{otherwise,} \end{cases} \tag{6}$$

where card(X) denotes cardinality of set X.

Note that the report time is unique, and it represents the cut-off point at which a vulnerability belongs to the database or not. So, the report time is chosen as the benchmark to rank the vulnerability database. The other time parameters, such as modified date, cannot rank the database effectively (a vulnerability possibly might not have a modified date, for example).

Figure 2 shows the frequency curves of the six basic metrics at time point $t = 2012$ for different values. For the exploitability metrics, the three curves of AV = Network, AC = Low, and AU = None have a higher position in their coordinate systems. This shows that a vulnerability v falling into the group (AV = Network, AC = Low, AU = None) is more vulnerable. Overall, the curves of exploitability metrics are divergent, while the curves of impact metrics are convergent.

Similarly, for $\forall (m_1, a_1), (m_2, a_2) \in \Omega$, and $m_1 \neq m_2$, the frequency of $(m_1 = a_1, m_2 = a_2)$ at time t can be calculated as follows:

$$f(m_1 = a_1, m_2 = a_2, t) = \frac{\text{card}(\mathscr{V}(m_1 = a_1, m_2 = a_2, t))}{\text{card}(\mathscr{V}(t))}, \tag{7}$$

where $\mathscr{V}(m_1 = a_1, m_2 = a_2, t) = \mathscr{V}(m_1 = a_1, t) \cap \mathscr{V}(m_2 = a_2, t)$. If $\mathscr{V}(t) = \emptyset$, $f(m_1 = a_1, m_2 = a_2, t) = 0$.

Table 1: Possible values of CVSS base metrics.

AV	0.395 (local)	0.646 (Adj.Net)	1 (network)
AC	0.35 (high)	0.61 (medium)	0.71 (low)
AU	0 (multiple)	0.56 (single)	0.704 (none)
CI, II, AI	0 (none)	0.275 (partial)	0.66 (complete)

Table 2: Input data for Chi-square analysis.

AV	AC	Count	Percentage
Local	High	471	0.87%
Local	Medium	1207	2.22%
Local	Low	5463	10.04%
Adj.Net	High	27	0.05%
Adj.Net	Medium	85	0.16%
Adj.Net	Low	146	0.27%
Network	High	2104	3.87%
Network	Medium	15796	29.02%
Network	Low	29133	53.52%

Table 3: Chi-square tests.

	Value	df	Asymp. Sig. (2-sided)
Pearson Chi-square	833.805[a]	4	.000
Likelihood ratio	909.498	4	.000
N of valid cases	54432		

[a]0 cells (0.0%) have expected count less than 5. The minimum expected count is 12.33.

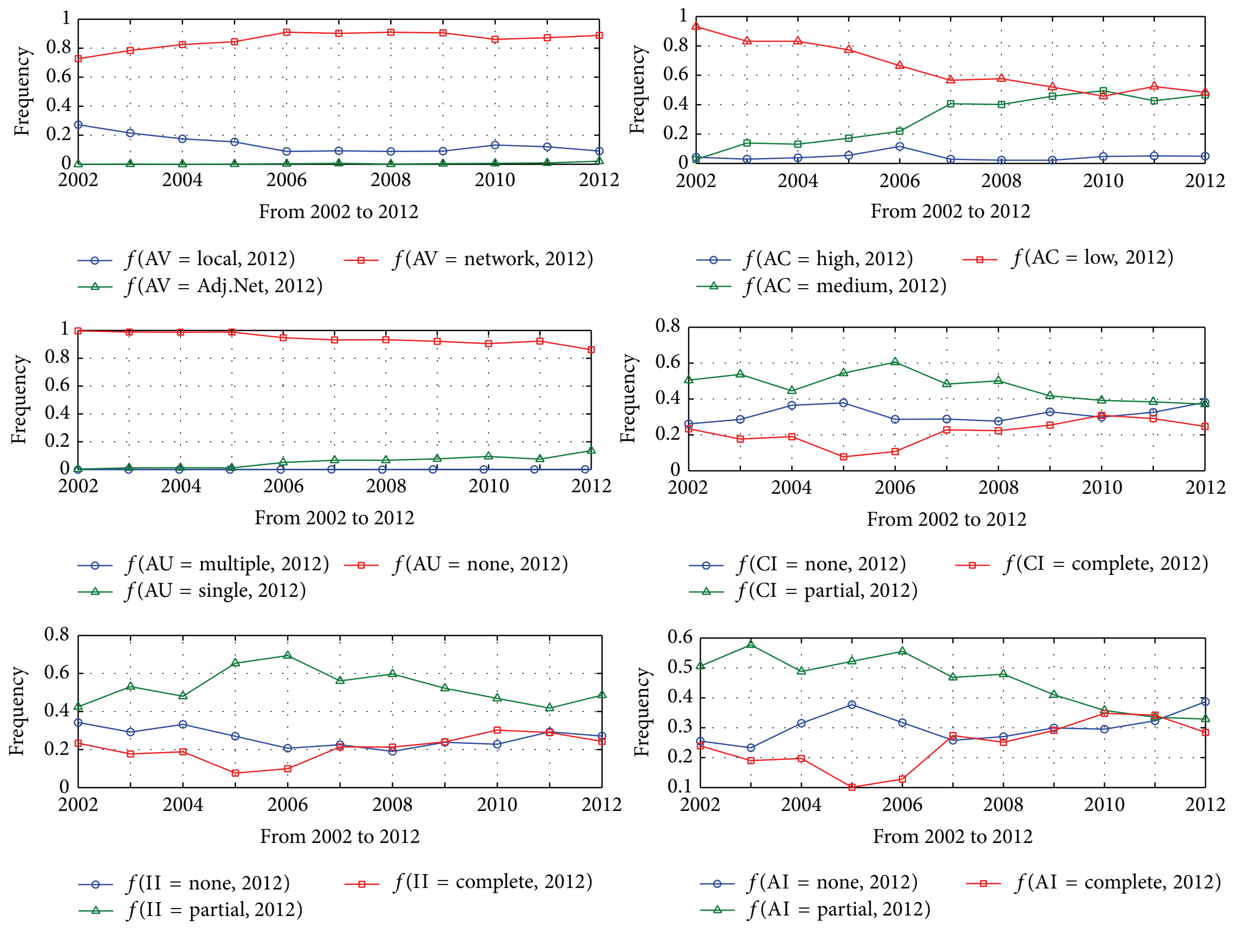


FIGURE 2: The frequency change curves of the six metrics from 2002 to 2012, where $T = \{2002, 2003, \ldots\}$. As can be seen, the curves of AV, AC, and AU are divergent, while the curves of CI, II, and AI have approximate convergence (to the value 1/3). These curves reflect the probability of the occurrence of the metric values of each metric.

2.2. Exploitability Score $E(v,t)$. Consider the correlations among AV, AC, and AU; the equation for exploitability subscore $E(v, t)$ is defined by the following probability:

$$E(v,t) = C(t) \cdot \Pr(v.\text{AV}, v.\text{AC}, v.\text{AU}, t), \tag{8}$$

where $\Pr(\cdot)$ is the probability measure. In contrast to v2, we use probability to define the exploitability score instead of the magic numbers in Table 1. The probability also includes the time point t as its parameter, and it can be given by the root of $n(t)$ solution of its frequency:

$$\Pr(\text{AV}, \text{AC}, \text{AU}, t) = \sqrt[n(t)]{f(\text{AV}, \text{AC}, \text{AU}, t)}. \tag{9}$$

Because the frequencies of the 27 vulnerability types of exploitability metrics are not of the same order of magnitude, $\sqrt[n(t)]{f(\text{AV}, \text{AC}, \text{AU}, t)}$ is used to approximate the probability $\Pr(\cdot)$ instead of $f(\text{AV}, \text{AC}, \text{AU}, t)$. So, there must exist the smallest positive integer n_0 such that

$$\sqrt[n_0]{\min\{f(\text{AV}, \text{AC}, \text{AU}, t) > 0\}} \geq \frac{1}{27}. \tag{10}$$

The basic idea of this equation is that the minimum and nonzero value $\min\{f(\text{AV}, \text{AC}, \text{AU}, t) > 0\}$ is close to 1/27 when the range $[0, 1]$ is divided into 27 classes. Since these subscores are normalized to the range $[0.0, 10.0]$, the coefficient of (8) can be determined by

$$C(t) = \frac{10}{\sqrt[n_0]{\max f(\text{AV}, \text{AC}, \text{AU}, t)}}. \tag{11}$$

For the database $\mathscr{V}(2012)$, the number of each exploitability type is listed in fourth column of Table 4. As can be seen, the value $\min\{f(\text{AV}, \text{AC}, \text{AU}, 2012) > 0\} = 1/54432$ and $\sqrt[4]{1/54432} = 0.0655 > 1/27$, so, $n_0 = 4$. Note that $\max\{f(\text{AV}, \text{AC}, \text{AU}, 2012)\} = 27419/54432$, and

$$C(2012) = \frac{10}{\sqrt[4]{27419/54432}} = 11.87. \tag{12}$$

From (8), the exploitability subscores of SVRA can be calculated; the results are listed in the fifth column of Table 4. CVSS v2 has 9 exploitability vectors with a score of 0, while SVRA has only one. The theoretical distributions of exploitability subscores for both SVRA and CVSS v2 are shown in Figure 3. From a theoretical viewpoint, v2 subscores have much less diversity than SVRA subscores. Figure 4

Table 4: Theoretical exploitability score comparison.

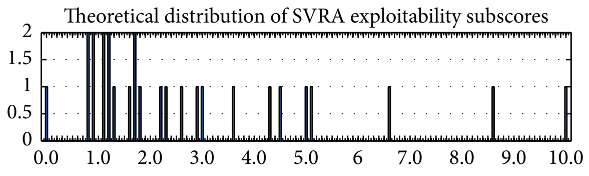
AV	AC	AU	Count[a]	$E(v,t)$	E (v2)
Local	High	Multiple	4	1.1	0
Local	High	Single	30	1.8	1.5
Local	High	None	437	3.6	1.9
Local	Medium	Multiple	2	0.9	0
Local	Medium	Single	83	2.3	2.7
Local	Medium	None	1122	4.5	3.4
Local	Low	Multiple	2	0.9	0
Local	Low	Single	193	2.9	3.1
Local	Low	None	5268	6.6	3.9
Adj.Net.	High	Multiple	1	0.8	0
Adj.Net.	High	Single	4	1.1	2.5
Adj.Net.	High	None	22	1.7	3.2
Adj.Net.	Medium	Multiple	0	0	0
Adj.Net.	Medium	Single	19	1.6	4.4
Adj.Net.	Medium	None	66	2.2	5.5
Adj.Net.	Low	Multiple	1	0.8	0
Adj.Net.	Low	Single	21	1.7	5.1
Adj.Net.	Low	None	124	2.6	6.5
Network	High	Multiple	7	1.3	0
Network	High	Single	213	3.0	3.9
Network	High	None	1884	5.1	4.9
Network	Medium	Multiple	5	1.2	0
Network	Medium	Single	902	4.3	6.8
Network	Medium	None	14889	8.6	8.6
Network	Low	Multiple	6	1.2	0
Network	Low	Single	1708	5.0	8.0
Network	Low	None	27419	10.0	10.0

[a] A total of 54432 vulnerabilities in $\mathscr{V}(2012)$ at $t = 2012$.

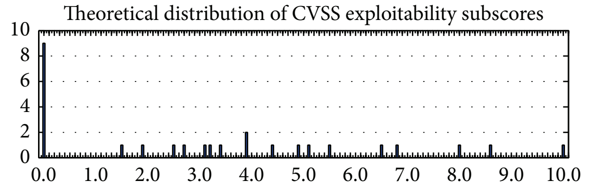
Table 5: Values of $n(t)$ and $C(t)$ from 2002 to 2012.

Year	$n(t)$	$C(t)$	Card($\mathscr{V}(t)$)	Mean($E(v,t)$)
2002	3	11.39	6662	3.2
2003	3	11.44	8157	2.8
2004	3	11.45	10796	2.7
2005	3	11.53	15409	2.7
2006	3	11.72	22389	2.6
2007	4	11.42	28823	3.1
2008	4	11.52	35808	3.1
2009	4	11.62	40655	3.1
2010	4	11.73	45516	3.1
2011	4	11.79	49724	3.0
2012	4	11.87	54432	2.9

shows how exploitability subscores change over time for two vulnerability types: (AV = Network, AC = High, AU = Single) and (AV = Local, AC = Low, AU = None). For SVRA, the exploitability subscore may increase, decrease, or remain unchanged. Table 5 lists the $n(t)$ and $C(t)$ values from 2002 to 2012. When the amount of vulnerability data $\mathscr{V}(t)$ increases, the changes for $C(t)$ and $n(t)$ are minor. This



(a)



(b)

Figure 3: Theoretical distributions of exploitability subscores for SVRA (a) and CVSS v2 (b), where $T = \{2002; 2003, \ldots\}$ and $t = 2012$.

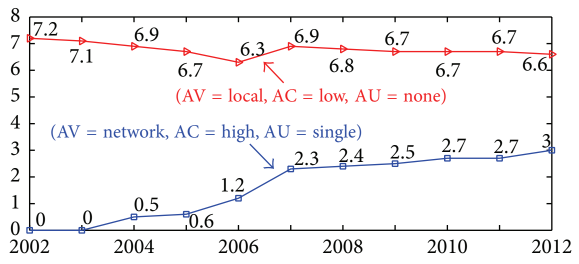


Figure 4: Exploitability subscore curves from 2002 to 2012 for two vulnerability types; the exploitability of a vulnerability changes when $\mathscr{V}(t)$ increases over time.

indicates that (8) has better stability and can dynamically compute $E(v,t)$ when $\mathscr{V}(t)$ changes over time.

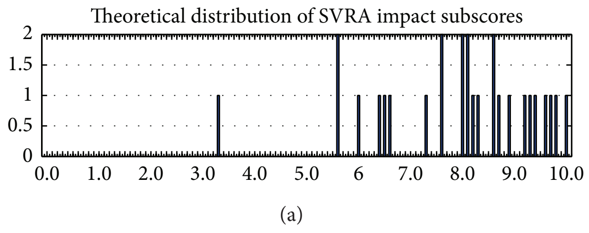
2.3. Impact Score $I(v,t)$. The impact caused by a vulnerability varies. Ideally, the sum of all categories of impact can be used to measure the impact subscore $I(v,t)$. However, there exist correlations among the impact metrics CI, II, and AI, so the equation for impact subscore is defined as

$$I(v,t) = D(t) \cdot \left[1 - \prod_{m \in S_i} (1 - \gamma(v.m,t))\right], \qquad (13)$$

where $S_i = \{\mathrm{CI}, \mathrm{II}, \mathrm{AI}\}$ and

$$\gamma(m,t) = \begin{cases} \frac{1}{3} f(m,t) & \text{if } m = \text{None} \\ \frac{1}{3} f(m,t) + 0.167 & \text{if } m = \text{Partial} \\ \frac{1}{3} f(m,t) + 0.333 & \text{if } m = \text{Complete}. \end{cases} \qquad (14)$$

From the frequency curves of CI, II, and AI in Figure 2, the approximate convergence value 1/3 is chosen as the



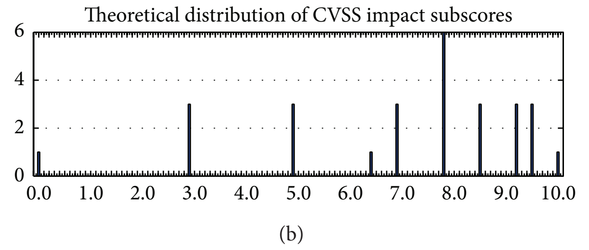


FIGURE 5: Theoretical distributions of impact subscores for SVRA (a) and CVSS v2 (b), where $T = \{2002, 2003, \ldots\}$ and $t = 2012$.

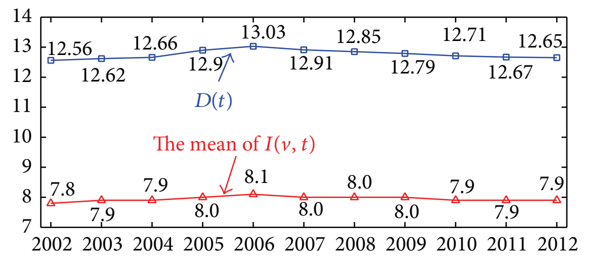


FIGURE 6: $D(t)$ and mean$(I(v,t))$ curves from 2002 to 2012; (13) has better stability and can dynamically compute $I(v,t)$ when $\mathscr{V}(t)$ changes.

coefficient of $\gamma(m,t)$. Let $\beta(v,t) = 1 - \prod_{m \in S_i}(1 - \gamma(v.m,t))$; this definition makes use of an idea similar to the inclusion-exclusion principle, and the parameter can be determined by the following equation:

$$D(t) = \frac{10}{\max \beta(v,t)}. \tag{15}$$

For database $\mathscr{V}(2012)$, the computing results for $\beta(v,t)$ are listed in the fourth column of Table 6. As can be seen, $\max \beta(v, 2012) = 0.7908$, so $D(2012) = 10/0.7908 = 12.65$. By (13), the impact subscores of SVRA can be calculated, and they are listed in the fifth column of Table 6. CVSS v2 has one impact vector (None, None, None) with a score of 0, while SVRA has none. For SVRA, the mean for the theoretical score is 7.9 and the median is 8.1; the standard deviation is 1.58 and the skew is −0.97. This represents a significant change from v2, which has a mean of 7.0, a median of 7.8, a standard deviation of 2.53, and a skew of −1.11. The theoretical distributions of the impact subscores for both SVRA and CVSS v2 are shown in Figure 5. From a theoretical viewpoint, v2 subscores have much less diversity than SVRA subscores.

Figure 6 shows the curves of the two variables $D(t)$ and mean$(I(v,t))$ from 2002 to 2012. When vulnerability data

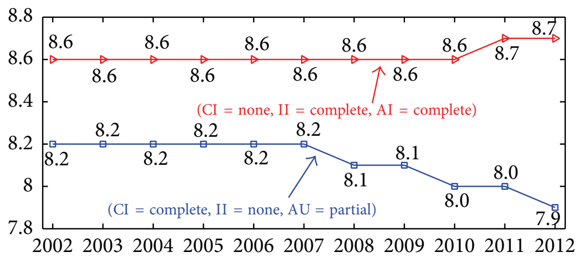


FIGURE 7: Impact subscore curves from 2002 to 2012 for two vulnerability types; the impact of a vulnerability changes when $\mathscr{V}(t)$ increases over time.

TABLE 6: Theoretical impact score comparison where $t = 2012$.

CI	II	AI	$\beta(v,t)$[a]	$I(v,t)$	I (v2)
None	None	None	0.2619	3.3	0.0
None	None	Partial	0.4412	5.6	2.9
None	None	Complete	0.5187	6.6	6.9
None	Partial	None	0.4734	6.0	2.9
None	Partial	Partial	0.6013	7.6	4.9
None	Partial	Complete	0.6566	8.3	7.8
None	Complete	None	0.5175	6.5	6.9
None	Complete	Partial	0.6347	8.0	7.8
None	Complete	Complete	0.6854	8.7	9.2
Partial	None	None	0.4453	5.6	2.9
Partial	None	Partial	0.5800	7.3	4.9
Partial	None	Complete	0.6382	8.1	7.8
Partial	Partial	None	0.6042	7.6	4.9
Partial	Partial	Partial	0.7004	8.9	6.4
Partial	Partial	Complete	0.7419	9.4	8.5
Partial	Complete	None	0.6374	8.1	7.8
Partial	Complete	Partial	0.7255	9.2	8.5
Partial	Complete	Complete	0.7635	9.7	9.5
Complete	None	None	0.5093	6.4	6.9
Complete	None	Partial	0.6285	8.0	7.8
Complete	None	Complete	0.6800	8.6	9.2
Complete	Partial	None	0.6499	8.2	7.8
Complete	Partial	Partial	0.7350	9.3	8.5
Complete	Partial	Complete	0.7717	9.8	9.5
Complete	Complete	None	0.6792	8.6	9.2
Complete	Complete	Partial	0.7572	9.6	9.5
Complete	Complete	Complete	0.7908	10.0	10.0

[a] $\beta(v,t) = 1 - \prod_{m \in S_i}(1 - \gamma(v.m,t))$.

$\mathscr{V}(t)$ increases, the changes for $D(t)$ and mean$(I(v,t))$ are minor. This indicates that (13) has better stability and can dynamically compute $I(v,t)$ when $\mathscr{V}(t)$ changes over time. Figure 7 shows how impact subscores change over time for two impact types: (CI = None, II = Complete, AI = Complete) and (CI = Complete, II = None, AU = Partial). For SVRA, the impact subscore can increase, decrease, or remain unchanged when $\mathscr{V}(t)$ changes over time.

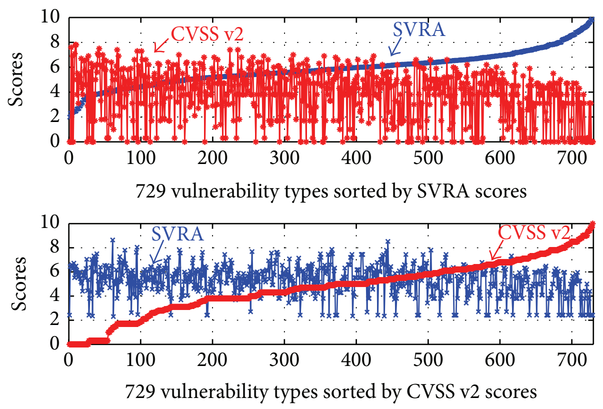


FIGURE 8: A theoretical score comparison of SVRA with CVSS v2 for 729 vulnerability types, where $T = \{2002, 2003, \ldots\}$ and $t = 2012$.

2.4. Base Score $B(v,t)$. Given time $t \in T$ and a vulnerability $v \in \mathscr{V}(t)$, the new equation of base score is as follows:

$$B(v,t) = \begin{cases} \dfrac{3I(v,t)}{5} + \dfrac{2E(v,t)}{5} & \text{if } I(v,t) \neq 0 \\ 0 & \text{otherwise.} \end{cases} \quad (16)$$

Figure 8 shows a comparison between SVRA and CVSS v2. Some vulnerability types have high SVRA scores but low v2 scores, and others have high v2 scores but low SVRA scores. We examined the theoretical distributions of SVRA and CVSS v2 scores; see Figure 9. For SVRA, the mean for the theoretical scores is 5.9, the median is 5.8, the standard deviation is 1.34, and the skew is 0.22. This represents a significant change from v2, which has a mean of 4.7, a median of 4.9, a standard deviation of 2.20, and a skew of −0.28. This illustrates that SVRA has superior numerical normality and stability. Figure 10 illustrates the changing trends of two vulnerability types (Type 1 and Type 2), as well as the mean of the SVRA base scores from 2002 to 2012. The CVSS v2 base scores of Type 1 and Type 2 are 6.6 and 5.6, respectively. However, the SVRA base score of Type 1 first decreases and then settles in the range $[7.7, 7.9]$. The $B(v,t)$ of Type 2 increases by 1.1 from 4.9 to 6.0. At first, the rise of the mean $\text{mean}(B(v,t))$ is rapid, and then the increase slows down. So, when new vulnerabilities are added into the database $\mathscr{V}(t)$, many of $\mathscr{V}(t)$ become more and more serious. In real life, two or more vulnerabilities may be combined to form a critical issue. The Google Chrome Pwnium full exploits are excellent examples, in which strings of vulnerabilities are combined into a full sandbox escape, resulting in arbitrary code execution. So, these curves reflect the fact that vulnerabilities interact with each other.

3. Experimental Analysis and Comparisons

This section describes our experimental analysis of the SVRA and CVSS v2 base scores for 54,432 vulnerabilities listed in CVE. Figure 11 shows a comparison between SVRA and CVSS v2. For the v2 experimental scores, the mean is 6.3, the median is 6.8, the standard deviation is 2.02, and the skew is −0.001. This represents an increase of 1.6 in the mean and 1.9 in the median from the theoretical data. Approximately 58.43% of the scores are above 5.0, 16.58% are at 5.0, and 24.98% are below 5.0. For the SVRA experimental scores, the mean is 8.1 and the median is 8.0. This represents an increase of 2.2 in both the mean and median from the theoretical data. The standard deviation is 1.26 and the skew is −0.39. Of the scores, approximately 99.09% are above 5.0, 0.21% are at 5.0, and 0.69% are below 5.0. This is consistent with the CVSS-SIG's goal to have the majority of scores above 5.0.

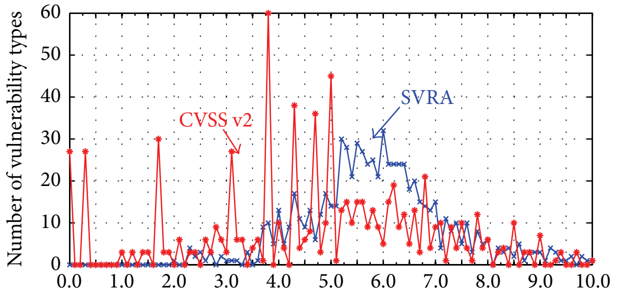


FIGURE 9: Theoretical distributions of SVRA and CVSS v2, where $T = \{2002, 2003, \ldots\}$ and $t = 2012$.

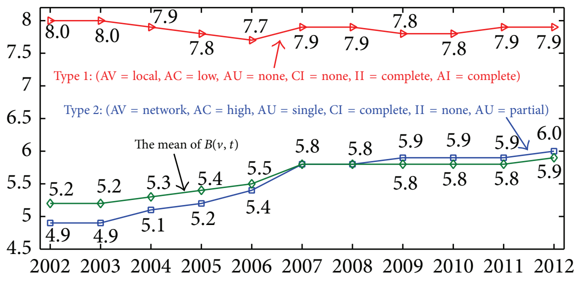


FIGURE 10: The base score curves from 2002 to 2012 for two vulnerability types and the mean of $B(v,t)$. This indicates the base score of a vulnerability changes when $\mathscr{V}(t)$ increases over time, where $T = \{2002, 2003, \ldots\}$.

The national vulnerability database (NVD) [4] generates a base score for each vulnerability and then assigns a ranking based on the score. The rankings are Low (0.0 to 3.9), Medium (4.0 to 6.9), and High (7.0 to 10.0) [10]. The motivation for having these rankings is to help organizations prioritize their mitigations of new vulnerabilities. Table 7 lists the comparison results of several vulnerabilities among the four authoritative security organizations (Secunia in Denmark, FrSIRT in France, ISS X-Force in the USA, and CVSS). As can be seen, SVRA coincides with the majority of the organizations. For the vulnerabilities CVE-2007-1497 and CVE-2007-2242, SVRA adjusts the CVSS rankings from High to Medium.

We also performed rankings for the theoretical data, as shown in Table 8. There exists a dramatic change in the CVSS v2 and SVRA, with SVRA having more Medium and High vulnerabilities but fewer Low vulnerabilities.

There are times when a vulnerability is scored as a 0.0 by CVSS v2 standards. These are often vulnerabilities that do

Table 7: Comparison of rating results.

Example	AV	AC	AU	CI	II	AI	Secunia	FrSIRT	X-Force	CVSS v2	SVRA
CVE-2007-1497	N	L	N	P	N	N	Medium (3/5)	Medium (2/4)	Medium (2/3)	High (3/3)	Medium (2/3)
CVE-2007-1754	N	M	N	C	C	C	High (4/5)	High (3/4)	High (3/3)	High (3/3)	High (3/3)
CVE-2007-1748	N	L	N	C	C	C	High (4/5)	Critical (4/4)	High (3/3)	High (3/3)	High (3/3)
CVE-2007-3338	N	L	N	C	C	C	Medium (3/5)	Critical (4/4)	High (3/3)	High (3/3)	High (3/3)
CVE-2007-3680	L	L	N	C	C	C	Low (2/5)	Medium (2/4)	High (3/3)	High (3/3)	High (3/3)
CVE-2007-2242	N	L	N	N	N	C	Medium (3/5)	Medium (2/4)	Low (1/3)	High (3/3)	Medium (2/3)

[a]"(m/n)" denotes the mth level of n severity levels; the bigger the m of a vulnerability, the higher severity level ranking of the vulnerability.

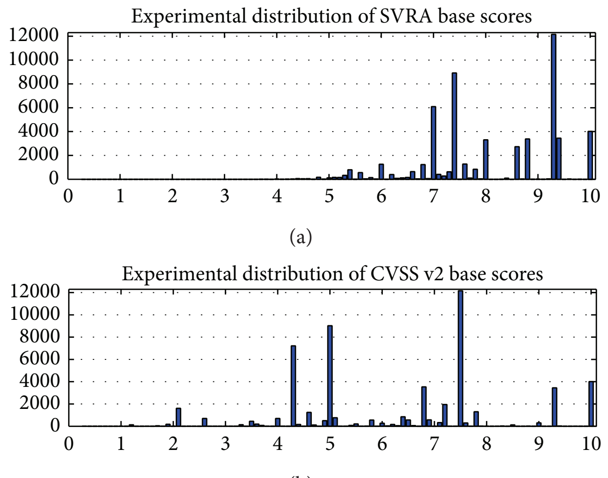


Figure 11: Experimental SVRA scores (a) and CVSS v2 scores (b), where $T = \{2002, 2003, \ldots\}$ and $t = 2012$.

Table 8: NVD severity rankings for theoretical data.

Rank	CVSS v2		SVRA	
	Count	Frequency	Count	Frequency
Low	251	34.43%	51	7.0%
Medium	370	50.75%	556	76.27%
High	108	14.82%	122	16.73%

pose some threat, albeit a limited one. Nevertheless, if the issue is considered a vulnerability by the industry, this should be reflected through the assignment of a real score. One such example is arbitrary site redirection. Per current CVSS v2 scoring rules, this would yield (AV = Network, AC = Medium, AU = None, CI = None, II = None, AI = None) with an SVRA score of 5.4.

4. Conclusions

The CVSS empirical values given by CVSS-SIG cannot distinguish software vulnerabilities that have identical scores but different severities. In this paper, a software vulnerability rating approach (SVRA) is proposed based on a vulnerability database. With the SVRA, the frequencies of CVSS metrics are analyzed at different times. The equations for both exploitability and impact subscores are given in terms of these frequencies. To create an SVRA score, SVRA performs a weighted average of these two subscores. As the frequency changes over time, each metric takes different values instead of the constant empirical value. The score of a vulnerability is dynamically computed at different time points using the vulnerability database. The theoretical and experimental results illustrate the efficiency of the SVRA.

Although the SVRA was developed for the base metric group, the approach can be extended to the temporal metric group and the environmental metric group. Further work will include predicting whether vulnerability severity changes so much over time that future modifications to the SVRA may be needed.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

This work was supported by the Funds NSFC61171121 and the Science Foundation of Chinese Ministry of Education—China Mobile 2012.

References

[1] M. Bozorgi, L. K. Saul, S. Savage, and G. M. Voelker, "Beyond heuristics: learning to classify vulnerabilities and predict exploits," in *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '10)*, pp. 105–113, July 2010.

[2] N. Poolsappasit, R. Dewri, and I. Ray, "Dynamic security risk management using Bayesian attack graphs," *IEEE Transactions on Dependable and Secure Computing*, vol. 9, no. 1, pp. 61–74, 2012.

[3] Forum of Incident Response and Security Teams, "CVSS Adopters," http://www.first.org/cvss/eadopters.html.

[4] National Institute of Standards and Technology, "National vulnerability database," http://nvd.nist.gov/.

[5] P. Mell, K. Scarfone, and S. Romanosky, "Common vulnerability scoring system (CVSS)," 2011, http://www.first.org/cvss/cvss-guide.html.

[6] K. Scarfone and P. Mell, "An analysis of CVSS version 2 vulnerability scoring," in *Proceedings of the 3rd International Symposium on Empirical Software Engineering and Measurement (ESEM '09)*, pp. 516–525, October 2009.

[7] National Infrastructure Advisory Council, "Common vulnerability scoring system," 2004, http://www.first.org/cvss/cvss-dhs-12-02-04.pdf.

[8] G. Reid, P. Mell, and K. Scarfone, "Cvss-sig version 2 history," Forum of Incident Response and Security Teams, June 2009, http://www.first.org/cvss/history.html.

[9] MITRE Corporation, "Common vulnerabilities and exposures (cve)," August 2009, http://cve.mitre.org/.

[10] National Institute of Standards and Technology, "National vulnerability database cvss scoring," August 2009, http://nvd.nist.gov/cvss.cfm.

Hindawi Publishing Corporation
Journal of Applied Mathematics
Volume 2014, Article ID 704231, 7 pages
http://dx.doi.org/10.1155/2014/704231

Research Article

A New Subband Adaptive Filtering Algorithm for Sparse System Identification with Impulsive Noise

Young-Seok Choi

Department of Electronic Engineering, Gangneung-Wonju National University, Gangneung 210-702, Republic of Korea

Correspondence should be addressed to Young-Seok Choi; yschoi@gwnu.ac.kr

Received 13 March 2014; Revised 21 April 2014; Accepted 5 May 2014; Published 20 May 2014

Academic Editor: Guiming Luo

This paper presents a novel subband adaptive filter (SAF) for system identification where an impulse response is sparse and disturbed with an impulsive noise. Benefiting from the uses of l_1-norm optimization and l_0-norm penalty of the weight vector in the cost function, the proposed l_0-norm sign SAF (l_0-SSAF) achieves both robustness against impulsive noise and remarkably improved convergence behavior more than the classical adaptive filters. Simulation results in the system identification scenario confirm that the proposed l_0-norm SSAF is not only more robust but also faster and more accurate than its counterparts in the sparse system identification in the presence of impulsive noise.

1. Introduction

Adaptive filtering algorithms have gained popularity and proven to be efficient in various applications such as system identification, channel equalization, and echo cancellation [1–4]. The normalized least mean square (NLMS) algorithm has become one of the most popular and widely used adaptive filtering algorithms because of its simplicity and robustness. Despite these advantages, the use of NLMS has been limited since it converges poorly for correlated input signals [2]. To address this problem, various approaches have been presented, such as the recursive least squares algorithm [2], the affine projection algorithm [2], and subband adaptive filtering (SAF) [5–9]. Among these, the SAF algorithm allocates the input signals and desired response into almost mutually exclusive subbands. This prewhitening characteristic of SAF allows each subband to converge almost separately so that the subband algorithms obtain faster convergence behavior. On the basis of these characteristics, Lee and Gan proposed a normalized SAF (NSAF) algorithm in [8, 9]. This work improves the convergence speed, while using almost the same computational complexity as the NLMS algorithm. However, the NSAF still suffers from the degradation of convergence performance in cases when an underlying system to be identified is sparse such as network echo path [10], underwater channel [11], and digital TV transmission channel [12]. Motivated by the proportionate step-size adaptive filtering [13, 14], the proportionate NSAF (PNSAF) has been presented to combat poor convergence in sparse system identification [15]. However, it does not exploit the sparsity condition itself. Moreover, the NSAF and PNSAF algorithms are highly sensitive to impulsive interference, leading to deteriorated convergence behavior. Impulsive interference exits in various applications such as acoustic echo cancellation [16], network cancellation [17], and subspace tracking [18].

To address the robustness issue, the sign SAF (SSAF) [19] has been developed based on the l_1-norm optimization, making it robust against impulsive interference. However, its use is limited in case of sparse system identification. Moreover, the SSAF converges poorly and fails to accelerate the convergence rate with the number of subbands.

In recent years, motivated by compressive sensing framework [20, 21] and the least absolute shrinkage and selection operator (LASSO) [22], a variety of adaptive filtering algorithms which incorporate the sparsity of a system have been developed unlike the proportionate adaptive filtering approach [23–27]. Along this line, the SAF with the l_1-norm penalty has been recently presented as an alternative for incorporating the sparsity of a system [28]. In particular, the l_0-norm of a system is able to represent the actual sparsity [24–26]. In this paper, a l_0-norm constraint SSAF (l_0-SSAF) is

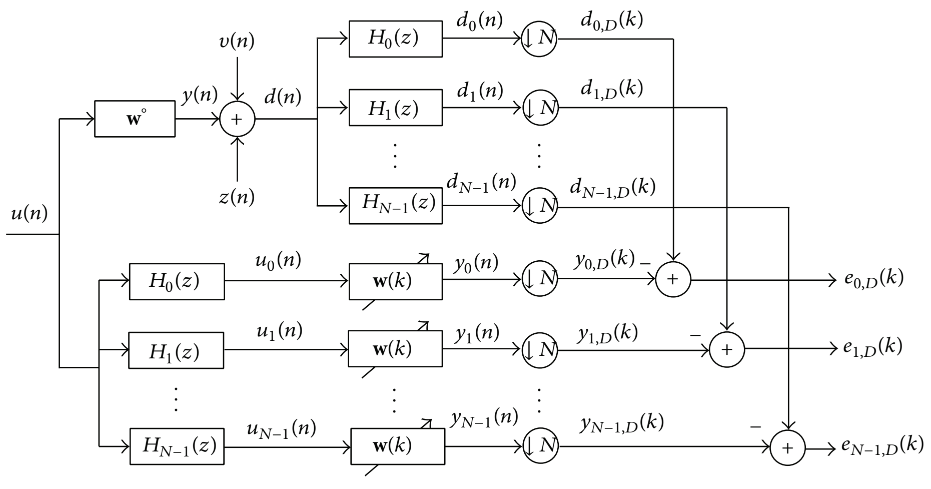


Figure 1: Subband structure used in the proposed SAF.

presented, aiming at developing a sparsity-aware SSAF. With this in mind, by integrating the l_0-norm penalty of the current weight vector into the l_1-norm optimization criterion, the l_0-SSAF benefits both superior convergence for sparse system identification and robustness against impulsive noise. In addition, the l_0-SSAF is derived from a l_1-norm optimization of the *a priori* error instead of the *a posteriori* error used in the SSAF. Thus, there is no need to approximate the *a posteriori* error with the *a priori* error to derive the update recursion of the l_0-SSAF. Simulation results show that the l_0-SSAF is superior to the conventional SAFs in identifying a sparse system in the presence of severe impulsive noise.

The remainder of the paper is organized as follows. Section 2 introduces the classical SAFs, followed by the derivation of the proposed l_0-SSAF algorithm in Section 3. Section 4 illustrates the computer simulation results and Section 5 concludes this study.

2. Conventional SAFs

Consider a desired signal $d(n)$ that arises from the system identification model

$$d(n) = \mathbf{u}(n)\mathbf{w}^{\circ} + v(n), \quad (1)$$

where $\mathbf{w}^{\circ}$ is a column vector for the impulse response of an unknown system that we wish to estimate, $v(n)$ accounts for measurement noise with zero mean and variance σ_v^2, and $\mathbf{u}(n) = [u(n)u(n-1)\cdots u(n-M+1)]$ is a $1\times M$ input vector.

Figure 1 shows the structure of the NSAF, where the desired signal $d(n)$ and output signal $u(n)$ are partitioned into N subbands by the analysis filters $H_0(z), H_1(z), \ldots, H_{N-1}(z)$. The resultant subband signals, $d_i(n)$ and $y_i(n)$ for $i = 0, 1, \ldots, N-1$, are critically decimated to a lower sampling rate commensurate with their bandwidth. Here, the variables n to index the original sequences and k to index the decimated sequences are used for all signals. Then, the decimated desired signal and the decimated filter output signal at each subband are defined as $d_{i,D}(k) = d_i(kN)$ and $y_{i,D}(k) = \mathbf{u}_i(k)\mathbf{w}(k)$, where $\mathbf{u}_i(k)$ is the input data vector for the ith subband such that

$$\mathbf{u}_i(k) = [u_i(kN), u_i(kN-1), \ldots, u_i(kN-M+1)] \quad (2)$$

and $\mathbf{w}(k) = [w_0(k), w_1(k), \ldots, w_{M-1}(k)]^T$ denotes an estimate for $\mathbf{w}^{\circ}$. Then, the decimated subband error vector is given by

$$e_{i,D}(k) = d_{i,D}(k) - y_{i,D}(k) = d_{i,D}(k) - \mathbf{u}_i(k)\mathbf{w}(k). \quad (3)$$

In [8], the authors have presented that the update recursion of the NSAF algorithm is given by

$$\mathbf{w}(k+1) = \mathbf{w}(k) + \mu \sum_{i=0}^{N-1} \frac{\mathbf{u}_i^T(k)}{\|\mathbf{u}_i(k)\|^2} e_{i,D}(k), \quad (4)$$

where μ is a step-size parameter. Then, the estimation error in all the N subbands, that is, $\mathbf{e}_D(k) = [e_{0,D}(k), \ldots, e_{N-1,D}(k)]^T$, can be written in a compact form as

$$\mathbf{e}_D(k) = \mathbf{d}_D(k) - \mathbf{U}(k)\mathbf{w}(k), \quad (5)$$

where the $N\times M$ subband data matrix $\mathbf{U}(k)$ and the $N\times 1$ desired response vector $\mathbf{d}(k)$ are given by

$$\begin{aligned} \mathbf{U}(k) &= [\mathbf{u}_0(k), \mathbf{u}_1(k), \ldots, \mathbf{u}_{N-1}(k)]^T, \\ \mathbf{d}_D(k) &= [d_{0,D}(k), d_{1,D}(k), \ldots, d_{N-1,D}(k)]^T. \end{aligned} \quad (6)$$

More recently, the SSAF [19] has been obtained from the following optimization criterion:

$$\begin{aligned} &\min_{\mathbf{w}(k+1)} && \|\mathbf{d}_D(k) - \mathbf{U}(k)\mathbf{w}(k+1)\|_1 \\ &\text{subject to} && \|\mathbf{w}(k+1) - \mathbf{w}(k)\|_2^2 \le \mu^2, \end{aligned} \quad (7)$$

where $\|\cdot\|_1$ denotes the l_1-norm and μ^2 is a parameter which prevents the weight coefficient vectors from abrupt change.

Using Lagrange multipliers to solve the constrained optimization problem and utilizing the accessible $\mathbf{e}_D(k)$ instead of unavailable *a posteriori* error, that is, $\mathbf{d}_D(k) - \mathbf{U}(k)\mathbf{w}(k+1)$, the update recursion of the SSAF is formulated as

$$\mathbf{w}(k+1) = \mathbf{w}(k) + \mu \frac{\mathbf{U}^T(k)\,\mathrm{sgn}\left[\mathbf{e}_D(k)\right]}{\sqrt{\sum_{i=0}^{N-1} \mathbf{u}_i(k)\,\mathbf{u}_i^T(k) + \delta}}, \quad (8)$$

where δ is a regularization parameter and $\mathrm{sgn}(\cdot)$ denotes the sign function, where $\mathrm{sgn}[\mathbf{e}_D(k)] = [\mathrm{sgn}(e_{0,D}(k)), \ldots, \mathrm{sgn}(e_{N-1,D}(k))]^T$.

3. Proposed l_0-Norm Constraint SSAF (l_0-SSAF)

Our objective is to cope with the sparsity of an underlying system while inheriting robustness from the l_1-norm optimization criterion. Our approach is to find a new weight vector, $\mathbf{w}(k+1)$, that minimizes the l_1-norm of the *a priori* error vector with the l_0-norm penalty of the current weight vector $\mathbf{w}(k)$ as follows:

$$\min_{\mathbf{w}} J(k) \triangleq \min_{\mathbf{w}} \left[\left\|\mathbf{e}_D(k)\right\|_1 + \gamma\|\mathbf{w}(k)\|_0\right], \quad (9)$$

where $\|\cdot\|_0$ denotes the l_0-norm and $\gamma(>0)$ is a regularization parameter which governs the compromise between the effect of the l_0-norm penalty term and the error vector related term. Note that the *a priori* error $\mathbf{e}_D(k)$ is used unlike the SSAF, leading to no approximation of the *a posteriori* error with the *a priori* error.

Taking derivative of $J(k)$, with respect to $\mathbf{w}(k)$, it leads to

$$\begin{aligned}\nabla_{\mathbf{w}(k)} J(k) &= -\mathbf{U}^T(k)\,\mathrm{sgn}\left(\mathbf{e}_D(k)\right) + \gamma \frac{\partial\|\mathbf{w}(k)\|_0}{\partial \mathbf{w}(k)} \\ &\triangleq -\mathbf{U}^T(k)\,\mathrm{sgn}\left(\mathbf{e}_D(k)\right) + \gamma \mathbf{f}_\beta(\mathbf{w}(k)),\end{aligned} \quad (10)$$

where $\mathbf{f}_\beta(\mathbf{w}(k)) \triangleq [f_\beta(w_0(k)), f_\beta(w_1(k)), \ldots, f_\beta(w_{M-1}(k))]^T$. To avoid a nonpolynomial hard problem from the l_0-norm minimization, the l_0-norm penalty is often approximated as follows [29]:

$$\|\mathbf{w}(k)\|_0 \approx \sum_{i=0}^{M-1}\left(1 - e^{-\beta|w_i(k)|}\right), \quad (11)$$

where the parameter β plays a role adjusting the degree of zero attraction. A mth component of the gradient for (11) is given by

$$\begin{aligned}\frac{\partial\|\mathbf{w}(k)\|_0}{\partial w_m(k)} &= f_\beta(w_m(k)) \\ &= \beta\,\mathrm{sgn}\left[w_m(k)\right] e^{-\beta|w_m(k)|} \quad \forall 0 \le m \le M-1.\end{aligned} \quad (12)$$

To reduce the computational cost in (12), the first-order Taylor series expansion of the exponential function is employed:

$$e^{-\beta|x|} \approx \begin{cases} 1 - \beta|x|, & |x| \le \dfrac{1}{\beta} \\ 0, & \text{elsewhere.} \end{cases} \quad (13)$$

Then, a gradient (12) is computed as

$$f_\beta(w_m(k)) = \begin{cases} \beta^2 w_m(k) + \beta, & -\dfrac{1}{\beta} \le w_m(k) < 0 \\ \beta^2 w_m(k) - \beta, & 0 < w_m(k) \le \dfrac{1}{\beta} \\ 0, & \text{elsewhere.} \end{cases} \quad (14)$$

Finally, the update recursion of the l_0-SSAF is given by

$$\mathbf{w}(k+1) = \mathbf{w}(k) + \mu\mathbf{U}^T(k)\,\mathrm{sgn}\left(\mathbf{e}_D(k)\right) - \kappa\mathbf{f}_\beta(\mathbf{w}(k)), \quad (15)$$

where μ is the step-size parameter and $\kappa = \mu\gamma$.

4. Simulation Results

To validate the performance of the proposed l_0-SSAF, computer simulations are carried out in a system identification scenario in which the unknown system is randomly generated. The length of the unknown system is $M = 128$, where S of them are nonzero. The nonzero filter weights are positioned randomly and their values are taken from a Gaussian distribution $\mathcal{N}(0, 1/S)$. Here, the sparse systems of the sparsity $S = 4, 8, 16, 32$ are considered. The adaptive filter and the unknown system are assumed to have the same number of taps. The input signals $u(n)$ are obtained by filtering a white, zero mean, Gaussian random sequence through a first-order system,

$$G_1(z) = \frac{1}{1 - 0.9z^{-1}}, \quad (16)$$

or a second-order system,

$$G_2(z) = \frac{1 + 0.5z^{-1} + 0.8z^{-2}}{1 - 0.9z^{-1}}. \quad (17)$$

A measurement noise $v(n)$ with white Gaussian distribution is added to the system output $y(n)$ such that the signal-to-noise ratio (SNR) is 20 dB, where the SNR is defined as

$$\mathrm{SNR} = 10\log_{10}\left(\frac{E\left[y^2(n)\right]}{E\left[v^2(n)\right]}\right), \quad (18)$$

where $y(n) = \mathbf{u}(n)\mathbf{w}^\circ$. An impulsive noise $z(n)$ is added to the system output $y(n)$ with the signal-to-interference ratio (SIR) of -30 or -10 dB correspondingly. The impulsive noise is modeled by a Bernoulli-Gaussian (BG) distribution [16], which is obtained as the product of a Bernoulli distribution and a Gaussian one; that is, $z(n) = \omega(n)\eta(n)$, where $\omega(n)$ is a Bernoulli process with a probability mass function given by $P(\omega) = 1 - Pr$ for $\omega = 0$ and $P(\omega) = Pr$ for $\omega = 1$. In addition, $\eta(n)$ is an additive white Gaussian noise with zero mean and variance $\sigma_\eta^2 = 1000\sigma_y^2$. Here, $Pr = 0.01$ is used. In order to compare the convergence performance, the normalized mean square deviation (NMSD),

$$\text{Normalized MSD} = E\left[\frac{\left\|\mathbf{w}^\circ - \mathbf{w}(k)\right\|^2}{\|\mathbf{w}^\circ\|^2}\right], \quad (19)$$

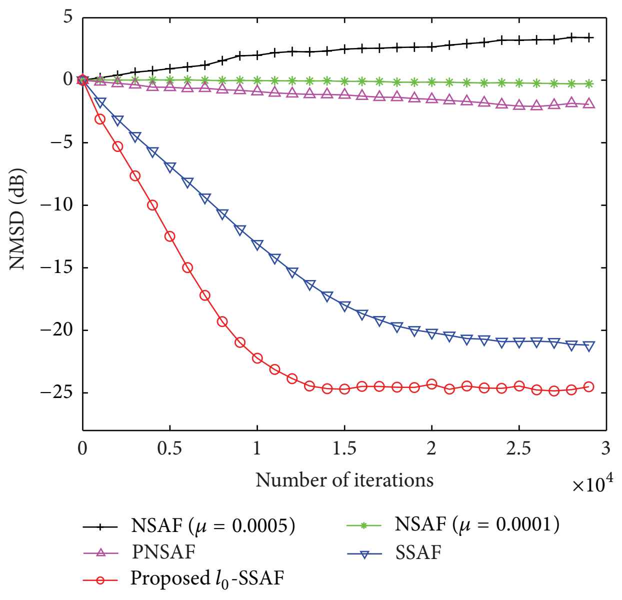


FIGURE 2: NMSD learning curves of the NSAF, PNSAF, SSAF, and l_0-SSAF algorithms [$N = 4$, SIR $= -30$ dB, input: Gaussian AR(1) with pole at 0.9].

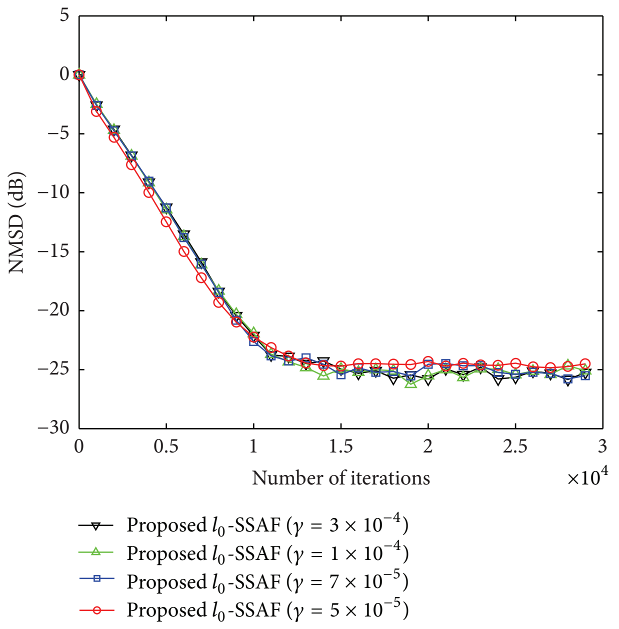


FIGURE 3: NMSD learning curves of the l_0-SSAF algorithm with various γ values [$N = 4$, SIR $= -30$ dB, input: Gaussian AR(1) with pole at 0.9].

is taken and averaged over 50 independent trials. The cosine-modulated filter banks [30] with the subband numbers of $N = 4$ are used in the simulations. The prototype filter of length $L = 32$ is used. The parameters used in simulations are as follows: NSAF ($\mu = 0.0005$ or 0.0001), SSAF ($\mu = 0.0005$, $\delta = 0.001$), PNSAF ($\mu = 0.0005$, $\rho = 0.01$), and l_0-SSAF ($\mu = 0.0003$, $\beta = 20$). The γ of the l_0-norm SSAF is obtained by repeated trials to minimize the steady-state NMSD. We use the input signals generated by $G_1(z)$ and $G_2(z)$ for Figures 2–7 and Figures 8 and 9, respectively.

Figure 2 shows the NMSD learning curves of the NSAF, PNSAF, SSAF, and l_0-norm SSAF algorithms in the case of SIR $= -30$ dB. For the l_0-SSAF, $\gamma = 5 \times 10^{-5}$ is chosen. Compared to the conventional SAF algorithms, the proposed l_0-SSAF yields remarkably improved convergence performance in terms of the convergence rate and the steady-state misalignment.

In Figure 3, to verify the effect of γ on convergence performance, the NMSD curves of the l_0-SSAF for different γ values are illustrated in the case of SIR $= -30$ dB. With different γ values ($\gamma = 3 \times 10^{-4}, 1 \times 10^{-4}, 7 \times 10^{-5}$, and 5×10^{-5}), the l_0-SSAF is not excessively sensitive to γ. The analysis for an optimal γ value remains a future work.

Figure 4 illustrates the NMSD learning curves of the NSAF, PNSAF, SSAF, and l_0-norm SSAF algorithms under SIR $= -10$ dB. The same γ value with Figure 2 is chosen. In the figure, similar results shown in Figure 2 are observed.

Figure 5 depicts the NMSD learning curves of the NSAF, PNSAF, SSAF, and l_0-SSAF algorithms for difference sparsity. Here, $S = 8, 16, 32$ were chosen. The same parameters as in Figure 2 are used. As can be seen, the more sparse the system, the better the convergence performance of the l_0-SSAF.

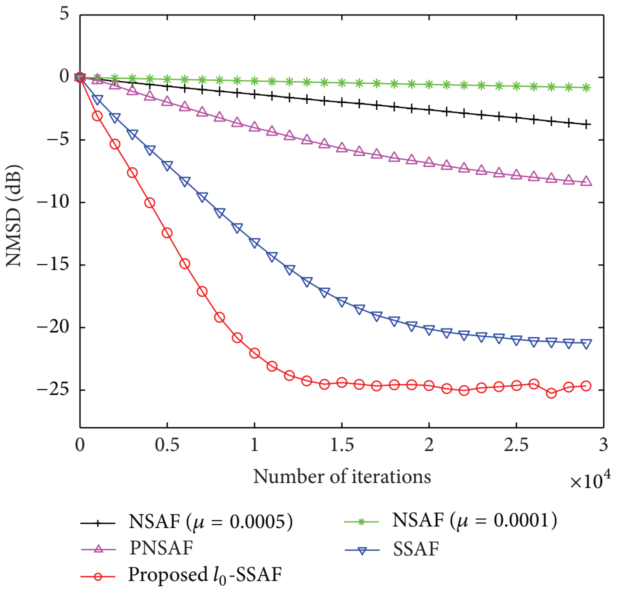


FIGURE 4: NMSD learning curves of the NSAF, PNSAF, SSAF, and l_0-SSAF algorithms [$N = 4$, SIR $= -10$ dB, input: Gaussian AR(1) with pole at 0.9].

Figure 6 shows the NMSD learning curves of the NSAF, PNSAF, SSAF, and l_0-SSAF algorithms with difference values of β in the case of $S = 4$. The values of $\beta = 1, 20, 50, 100$ were used. Also, the same step-size parameter $\mu = 0.0003$ is chosen. In the figure, it is apparent that the larger the value of β, the higher the steady-state MSD. However, the optimal value of β remains a future issue.

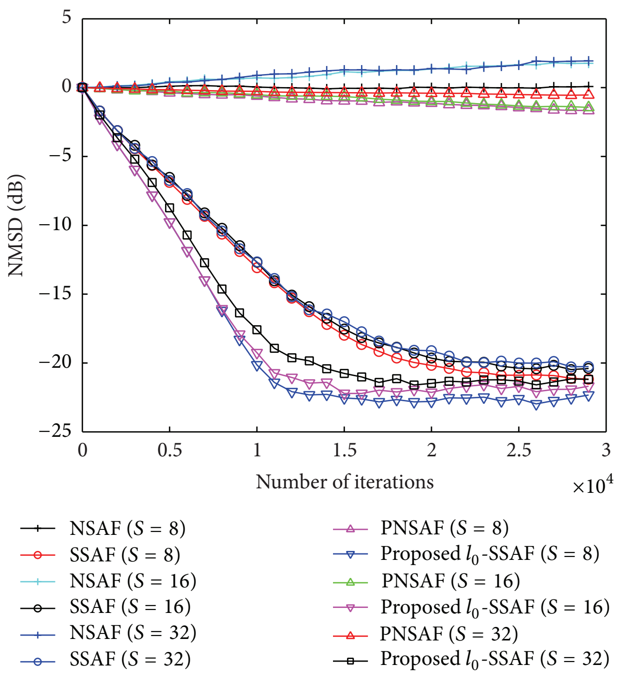


FIGURE 5: NMSD learning curves of the NSAF, PNSAF, SSAF, and l_0-SSAF algorithms for different sparsity (S = 8, 16, 32) [N = 4, SIR = −30 dB, input: Gaussian AR(1) with pole at 0.9].

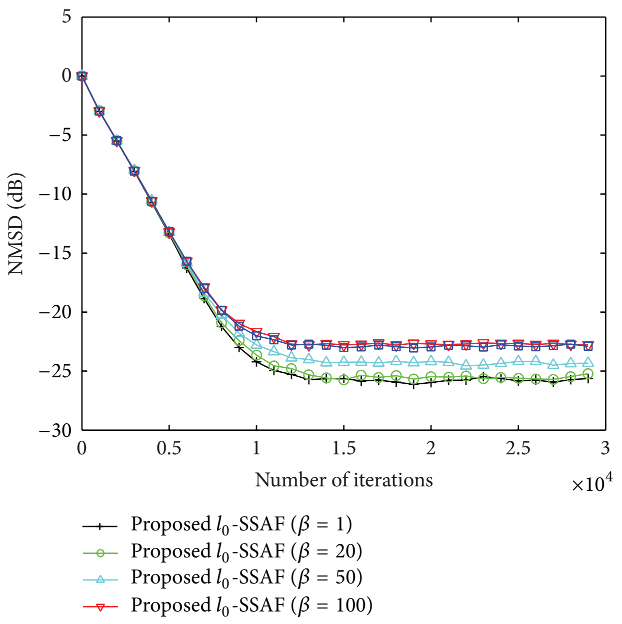


FIGURE 6: NMSD learning curves of the NSAF, PNSAF, SSAF, and l_0-SSAF algorithms with difference values of β (β = 1, 20, 50, 100) [N = 4, SIR = −30 dB, input: Gaussian AR(1) with pole at 0.9].

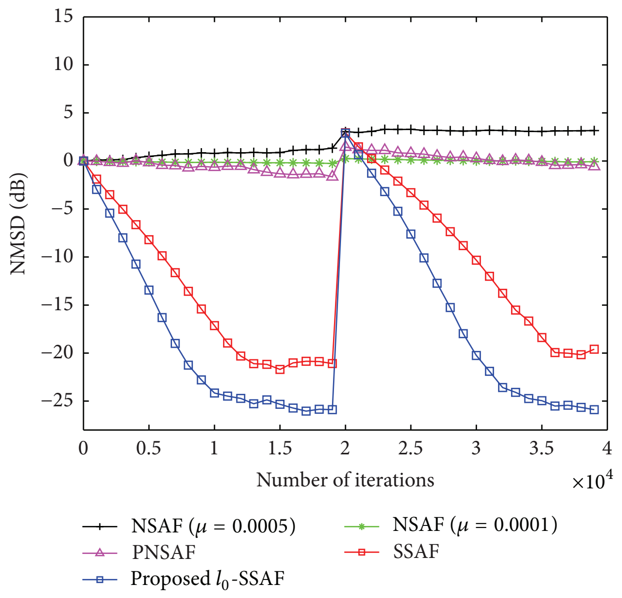


FIGURE 7: NMSD learning curves of the NSAF, PNSAF, SSAF, and l_0-SSAF algorithms in case of a time-varying unknown system (N = 4). The system is right-shifted for 20 taps at 20000th iteration [N = 4, SIR = −30 dB, input: Gaussian AR(1) with pole at 0.9].

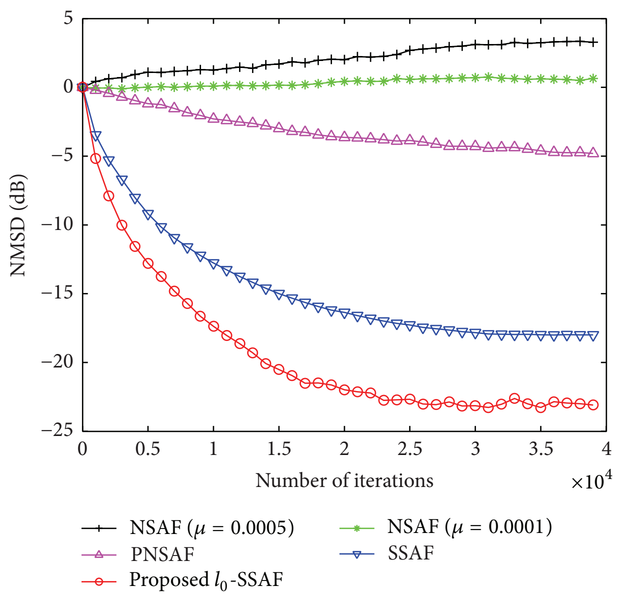


FIGURE 8: NMSD learning curves of the NSAF, PNSAF, SSAF, and l_0-SSAF algorithms [N = 4, SIR = −30 dB, input: Gaussian AR(2,2)].

Next, the tracking capabilities of the algorithms to a sudden change in the system are tested for SIR = −30 dB. Figure 7 shows the results in case when an unknown system is right-shifted for 20 taps. The same value of γ of Figure 2 is used. The figure shows that the l_0-SSAF keeps track of weight change while achieving a faster convergence rate and a low steady-state misalignment compared to the conventional SAF algorithms.

Finally, Figures 8 and 9 show the simulation results with the different input signal generated by $G_2(z)$ for SIR = −30 dB and −10 dB, respectively. The same parameters of all SAF algorithms in Figure 2 are chosen in Figures 8 and 9. We can see similar results in previous figures, implying the capability of the l_0-norm SSAF over the classical SAF algorithms for different input signal.

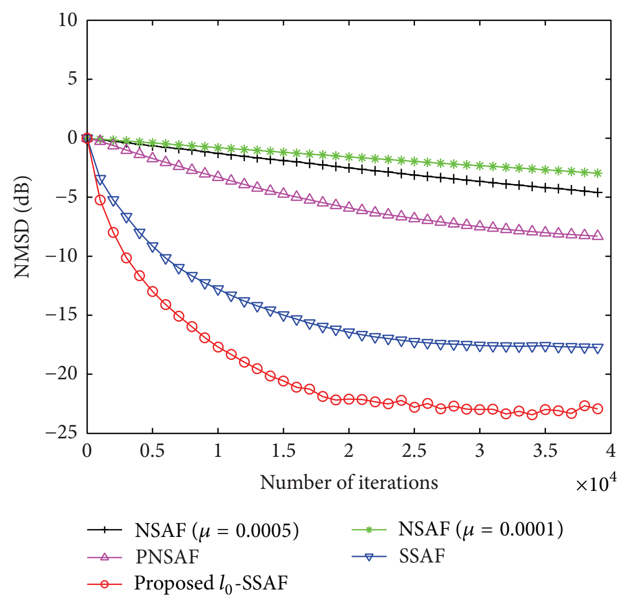


FIGURE 9: NMSD learning curves of the NSAF, PNSAF, SSAF, and l_0-SSAF algorithms [$N = 4$, SIR $= -10$ dB, input: Gaussian AR(2,2)].

5. Conclusion

This paper has proposed a robust and sparse-aware SSAF algorithm which incorporates the sparsity condition of a system into the l_1-norm optimization criterion of the *a priori* error vector. By utilizing the l_0-norm penalty of the current weight vector and approximating it to avoid a nonpolynomial hard problem, the update recursion of the proposed l_0-norm SSAF is obtained while reducing the computational cost using Taylor series expansion. The simulation results indicate that the proposed l_0-SSAF achieves highly improved convergence performance over the conventional SAF algorithms where a system is not only sparse but also disturbed with impulsive noise.

Conflict of Interests

The author declares that there is no conflict of interests regarding the publication of this paper.

References

[1] S. Haykin, *Adaptive Filter Theory*, Prentice Hall, Upper Saddle River, NJ, USA, 4th edition, 2002.

[2] A. H. Sayed, *Fundamentals of Adaptive Filtering*, John Wiley & Sons, New York, NY, USA, 2003.

[3] P. S. R. Diniz, *Adaptive Filtering: Algorithms and Practical Implementations*, Kluwer Academic, Boston, Mass, USA, 3rd edition, 2008.

[4] M. M. Sondhi, "The history of echo cancellation," *IEEE Signal Processing Magazine*, vol. 23, no. 5, pp. 95–102, 2006.

[5] A. Gilloire and M. Vetterli, "Adaptive filtering in subbands with critical sampling: analysis, experiments, and application to acoustic echo cancellation," *IEEE Transactions on Signal Processing*, vol. 40, no. 8, pp. 1862–1875, 1992.

[6] M. de Courville and P. Duhamel, "Adaptive filtering in subbands using a weighted criterion," *IEEE Transactions on Signal Processing*, vol. 46, no. 9, pp. 2359–2371, 1998.

[7] S. S. Pradhan and V. U. Redd, "A new approach to subband adaptive filtering," *IEEE Transactions on Signal Processing*, vol. 47, no. 3, pp. 655–664, 1999.

[8] K. A. Lee and W. S. Gan, "Improving convergence of the NLMS algorithm using constrained subband updates," *IEEE Signal Processing Letters*, vol. 11, no. 9, pp. 736–739, 2004.

[9] K. A. Lee and W. S. Gan, "Inherent decorrelating and least perturbation properties of the normalized subband adaptive filter," *IEEE Transactions on Signal Processing*, vol. 54, no. 11, pp. 4475–4480, 2006.

[10] D. L. Duttweiler, "Proportionate normalized least-mean-squares adaptation in echo cancelers," *IEEE Transactions on Speech and Audio Processing*, vol. 8, no. 5, pp. 508–518, 2000.

[11] W. Li and J. C. Preisig, "Estimation of rapidly time-varying sparse channels," *IEEE Journal of Oceanic Engineering*, vol. 32, no. 4, pp. 927–939, 2007.

[12] W. F. Schreiber, "Advanced television systems for terrestrial broad-casting: some problems and some proposed solutions," *Proceedings of IEEE*, vol. 83, no. 6, pp. 958–981, 1995.

[13] S. L. Gay, "Efficient, fast converging adaptive filter for network echo cancellation," in *Proceedings of the 32nd Asilomar Conference on Signals, Systems & Computers*, vol. 1, pp. 394–398, Pacific Grove, Calif, USA, November 1998.

[14] H. Deng and M. Doroslovački, "Improving convergence of the PNLMS algorithm for sparse impulse response identification," *IEEE Signal Processing Letters*, vol. 12, no. 3, pp. 181–184, 2005.

[15] M. S. E. Abadi, "Proportionate normalized subband adaptive filter algorithms for sparse system identification," *Signal Processing*, vol. 89, no. 7, pp. 1467–1474, 2009.

[16] L. R. Vega, H. Rey, J. Benesty, and S. Tressens, "A new robust variable step-size NLMS algorithm," *IEEE Transactions on Signal Processing*, vol. 56, no. 5, pp. 1878–1893, 2008.

[17] Z. Yang, Y. R. Zheng, and S. L. Grant, "Proportionate affine projection sign algorithms for network echo cancellation," *IEEE Transactions on Audio, Speech and Language Processing*, vol. 19, no. 8, pp. 2273–2284, 2011.

[18] B. Liao, Z. G. Zhang, and S. C. Chan, "A new robust Kalman filter-based subspace tracking algorithm in an impulsive noise environment," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 57, no. 9, pp. 740–744, 2010.

[19] J. Ni and F. Li, "Variable regularisation parameter sign subband adaptive filter," *Electronics Letters*, vol. 46, no. 24, pp. 1605–1607, 2010.

[20] D. L. Donoho, "Compressed sensing," *IEEE Transactions on Information Theory*, vol. 52, no. 4, pp. 1289–1306, 2006.

[21] E. J. Candès, J. Romberg, and T. Tao, "Robust uncertainty principles: exact signal reconstruction from highly incomplete frequency information," *IEEE Transactions on Information Theory*, vol. 52, no. 2, pp. 489–509, 2006.

[22] R. Tibshirani, "Regression shrinkage and selection via the lasso," *Journal of the Royal Statistical Society B: Methodological*, vol. 58, no. 1, pp. 267–288, 1996.

[23] Y. Chen, Y. Gu, and A. O. Hero, "Sparse LMS for system identification," in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP '09)*, pp. 3125–3128, Taipei, Taiwan, April 2009.

[24] Y. Gu, J. Jin, and S. Mei, "l_0 norm constraint LMS algorithm for sparse system identification," *IEEE Signal Processing Letters*, vol. 16, no. 9, pp. 774–777, 2009.

[25] J. Jin, Y. Gu, and S. Mei, "A stochastic gradient approach on compressive sensing signal reconstruction based on adaptive filtering framework," *IEEE Journal on Selected Topics in Signal Processing*, vol. 4, no. 2, pp. 409–420, 2010.

[26] E. M. Eksioglu and A. K. Tanc, "RLS algorithm with convex regularization," *IEEE Signal Processing Letters*, vol. 18, no. 8, pp. 470–473, 2011.

[27] N. Kalouptsidis, G. Mileounis, B. Babadi, and V. Tarokh, "Adaptive algorithms for sparse system identification," *Signal Processing*, vol. 91, no. 8, pp. 1910–1919, 2011.

[28] Y.-S. Choi, "Subband adaptive filtering with l_1-norm constraint for sparse system identification," *Mathematical Problems in Engineering*, vol. 2013, Article ID 601623, 7 pages, 2013.

[29] P. S. Bradley and O. L. Mangasarian, "Feature selection via concave minimization and support vector machines," in *Proceedings of the International Conference on Machine Learning (ICML '98)*, pp. 82–90, 1998.

[30] P. P. Vaidyanathan, *Multirate Systems and Filterbanks*, Prentice Hall, Englewood Cliffs, NJ, USA, 1993.

Hindawi Publishing Corporation
Journal of Applied Mathematics
Volume 2014, Article ID 789213, 12 pages
http://dx.doi.org/10.1155/2014/789213

Research Article

Robustness Analysis of Floating-Point Programs by Self-Composition

Liqian Chen, Jiahong Jiang, Banghu Yin, Wei Dong, and Ji Wang

National Laboratory for Parallel and Distributed Processing, National University of Defense Technology, Changsha 410073, China

Correspondence should be addressed to Liqian Chen; lqchen@nudt.edu.cn

Received 14 February 2014; Accepted 7 April 2014; Published 20 May 2014

Academic Editor: Xiaoyu Song

Robustness is a key property for critical systems that run in uncertain environments, to ensure that small input perturbations can cause only small output changes. Current critical systems often involve lots of floating-point computations which are inexact. Robustness analysis of floating-point programs needs to consider both the uncertain inputs and the inexact computation. In this paper, we propose to leverage the idea of self-composition to transform the robustness analysis problem into a reachability problem, which enables the use of standard reachability analysis techniques such as software model checking and symbolic execution for robustness analysis. To handle floating-point arithmetic, we employ an abstraction that encompasses the effect of rounding and that can encompass all rounding modes. It converts floating-point expressions into linear expressions with interval coefficients in exact real arithmetic. On this basis, we employ interval linear programming to compute the maximum output change or maximum allowed input perturbation for the abstracted programs. Preliminary experimental results of our prototype implementation are encouraging.

1. Introduction

Uncertainty and inexactness in computing have attracted much attention in computer science. In Cyber Physical Systems (CPS), the discrete world of computation is integrated with the continuous world of physical processes. Moreover, CPS run in the open environmental context and thus have to deal with uncertain data which may come from noisy sensor data or approximate computation. Hence, inputs for programs in CPS are of intrinsic uncertainty. On the other hand, due to finite precision on computers, physical values are truncated into digital ones. In modern computers, real numbers are approximated by a finite set of floating-point numbers. Due to the pervasive rounding errors, numerical computation using floating-point arithmetic is not exact. Since many safety-critical CPS systems (such as aircrafts, automobiles, and medical devices) often involve lots of numerical computations, there is a great need to ensure that these programs are *robust* with respect to the uncertain input as well as the inexact computation.

Although *robustness* is long known as a standard correctness property for control systems [1], considering the robustness of programs is quite recent [2–5]. Intuitively, robustness of a program means that small input perturbations of the program can cause only small output changes. Much existing work on analyzing robustness of programs assumes that the analyzed program is in exact real arithmetic, although floating-point computation is pervasive in practical applications. This paper targets the analysis of robustness properties of floating-point programs.

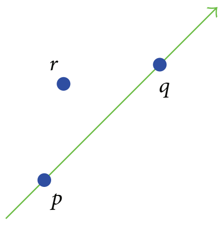
A program using floating-point arithmetic often exhibits more robustness issues than that using exact real arithmetic, due to the misunderstandings and nonintuitive behaviors of floating-point semantics. Although floating-point arithmetic is quite different from the exact real arithmetic, most developers of floating-point programs will write programs as if computations were done in exact arithmetic. For the same input, the control flow of the program using floating-point arithmetic can be different from the one that would be taken assuming exact real arithmetic. Similarly, for two inputs whose values are close to each other, the resulting two control flows of the same program can be different (even when following exact real arithmetic). Two different control flows may lead to very large difference in outputs.

We illustrate the robustness problem due to floating-point computation using a motivating example shown in Figure 1, which is a "classroom" example of a robustness problem frequently used in the field of geometric computations [6]. The program `Orientation` implements the 2D orientation test that decides whether a point r lies to the left of, to the right of, or on the line $\overrightarrow{pq}$ defined by the 2 points p, q, by evaluating the sign of a determinant `det` which is expressed in terms of the coordinates of the input points. Due to rounding errors, the floating-point computation of the determinant `det` may lead to a wrong result when the true determinant (via exact real arithmetic) is close to zero. From the robustness point of view, for this program, even a very small input perturbation may lead to an output change of 1 or 2. If the rounding modes for the floating-point operations are not determinate in the program, the output change can be 2 even when there is no perturbation in the inputs (by running the program in different rounding modes). This misinformation may then lead to a failure of a computational geometry application (e.g., crash or not terminate) or produce wrong results [6].

Analyzing robustness of floating-point programs is more challenging than analyzing programs assuming exact real arithmetic, since besides the input perturbations, we need to consider also the inexactness of floating-point computation. The floating-point program itself acts as if inputs were perturbated due to the pervasive rounding errors or nondeterminate rounding modes. There exist a few known pitfalls of analyzing and verifying floating-point programs [7].

In this paper, we present a robustness analysis method for floating-point programs. The key idea is to leverage the self-composition technique from the field of secure information flow to transform the robustness analysis problem into a reachability (safety) problem. Then we use standard rechability analysis techniques such as software model checking and symbolic execution to analyze the self-composed programs. To cope with floating-point arithmetic, we utilize a rounding mode insensitive abstraction method to abstract floating-point expressions into linear expressions with interval coefficients in the field of reals. On this basis, we use interval linear programming to compute the maximum output change (when given the input perturbation) or the maximum allowed input perturbation (when given the output change) for the abstracted programs. The preliminary experimental results are promising on benchmark programs.

The rest of the paper is organized as follows. Section 2 reviews the IEEE 754 floating-point arithmetic and the basic theory of interval linear systems as well as interval linear programming. Section 3 presents the robustness analysis approach via self-composition for programs (that assume exact real arithmetic). Section 4 presents the techniques



```
int Orientation (float px, float py, float qx, float qy, float rx, float ry)
{
    float pqx = qx − px, pqy = qy − py;
    float prx = rx − px, pry = ry − py;
    float det = pqx ∗ pry − pqy ∗ prx;
    if (det > 0) return 1;
    if (det < 0) return −1;
    return 0;
}
```

FIGURE 1: Floating-point implementation for orientation test of 2D points.

to handle floating-point arithmetic. Section 5 presents our prototype implementation together with preliminary experimental results. Section 6 discusses some related work before Section 7 concludes.

2. Preliminaries

In this section, we briefly provide the background on the IEEE 754 floating-point arithmetic and the basic theory on interval arithmetics as well as interval linear programming.

2.1. The IEEE 754 Floating-Point Arithmetic.

A digital computer cannot represent all possible real numbers in mathematics exactly. In computing, floating-point numbers provide an approach to represent a finite subset of the real numbers. In this paper, we focus on analyzing programs with respect to the binary formats of the IEEE 754 floating-point standard [8] which is the most commonly used floating-point representation and is followed by almost all modern computers.

In the IEEE 754 standard, the binary representation of a floating-point number x can be described as $x = (-1)^S \times M \times 2^E$, where

(i) S is the 1-bit *sign* of x, which represents that x is positive (when $S = 0$) or negative (when $S = 1$);

(ii) $E = e - \mathbf{bias}$ is called the *exponent*, where e is a biased $\mathbf{e}$-bit unsigned integer and $\mathbf{bias} = 2^{\mathbf{e}-1} - 1$;

(iii) $M = m_0.m_1m_2 \dots m_{\mathbf{p}}$ is called the *significand*, where $f = .m_1m_2 \dots m_{\mathbf{p}}$ represents a $\mathbf{p}$-bit fraction and m_0 is the hidden bit without need of storage.

The values of $\mathbf{e}$, $\mathbf{bias}$, $\mathbf{p}$ depend on the floating-point formats. The IEEE 754 standard supports several formats, among which the basic formats include

(i) 32-bit single-precision format, where $\mathbf{e} = 8$ (and thus $\mathbf{bias} = 127$), $\mathbf{p} = 23$;

(ii) 64-bit double-precision format, where $\mathbf{e} = 11$ (and thus $\mathbf{bias} = 1023$), $\mathbf{p} = 52$.

According to the value of e, the floating-point numbers can be divided into the following categories:

(i) *normalized* number $(-1)^S \times 1.f \times 2^{e-\mathbf{bias}}$, when $1 \le e \le 2^{\mathbf{e}} - 2$;

(ii) *denormalized* number $(-1)^S \times 0.f \times 2^{1-\mathbf{bias}}$, when $e = 0$ and $f \neq 0$;

(iii) $+0$ or -0, when $e = 0$ and $f = 0$;

(iv) $+\infty$ or $-\infty$, when $e = 2^{\mathbf{e}} - 1$ and $f = 0$;

(v) NaN (Not a Number), when $e = 2^{\mathbf{e}} - 1$ and $f \neq 0$.

Let $\mathbf{F}$ be the set of floating-point formats. For each $\mathbf{f} \in \mathbf{F}$, we define

(i) $mf_{\mathbf{f}} \overset{\text{def}}{=} 2^{1-\mathbf{bias}-\mathbf{p}}$, the smallest nonzero positive floating-point number;

(ii) $Mf_{\mathbf{f}} \overset{\text{def}}{=} (2 - 2^{-\mathbf{p}})2^{2^{\mathbf{e}}-\mathbf{bias}-2}$, the largest noninfinity floating-point number.

In general, the result of a floating-point operation may not be exactly representable in the floating-point representation, and thus the result needs to be rounded into a floating-point number. The IEEE 754 standard supports four rounding modes: toward nearest, toward $+\infty$, toward $-\infty$, and toward zero. In this paper, in order to distinguish floating-point arithmetic operations from exact real arithmetic ones, we introduce additional notations. As usual, $\{+, -, \times, /\}$ are used as exact rational arithmetic operations. The corresponding floating-point operations are denoted by $\{\oplus_{\mathbf{f},r}, \ominus_{\mathbf{f},r}, \otimes_{\mathbf{f},r}, \oslash_{\mathbf{f},r}\}$, tagged with a floating-point format $\mathbf{f} \in \mathbf{F}$ and a rounding mode $r \in \{+\infty, -\infty, 0, n\}$ (n representing rounding to nearest). We also use ? to denote arbitrary rounding mode.

Due to rounding errors, many well-known algebraic properties (such as associativity and distributivity) over the reals do not hold for floating-point arithmetic.

Example 1. Consider the following expressions in the 32-bit single-precision floating-point arithmetic:

$$\begin{gathered}\left(2^{24} \oplus_{32,?} - 2^{24}\right) \oplus_{32,?} 1 = 1 \\ \left(2^{24} \oplus_{32,-\infty} 1\right) \oplus_{32,-\infty} - 2^{24} = 0 \\ \left(2^{24} \oplus_{32,+\infty} 1\right) \oplus_{32,+\infty} - 2^{24} = 2.\end{gathered} \tag{1}$$

Note that in the 32-bit single-precision format, the significand is $M = m_0.m_1 m_2 \ldots m_{\mathbf{23}}$. However, to represent the exact result of $2^{24} + 1$ over the reals, we need one more bit for the significand M (say $m_{\mathbf{24}}$). Hence, rounding happens. $2^{24} \oplus_{32,-\infty} 1$ will result in 2^{24}, while $2^{24} \oplus_{32,+\infty} 1$ will result in $(1 + 2^{-23}) \times 2^{24}$.

2.2. Interval Linear Systems and Interval Linear Programming. Let $\underline{A}, \overline{A} \in \mathbb{R}^{m\times n}$ be two matrices with $\underline{A} \le \overline{A}$, where comparison operators are defined element-wise; then the set of matrices $\mathbf{A} \in \mathbb{IR}^{m\times n}$ defined by

$$\mathbf{A} = \left[\underline{A}, \overline{A}\right] = \left\{A \in \mathbb{R}^{m\times n} : \underline{A} \le A \le \overline{A}\right\} \tag{2}$$

is called an *interval matrix*, and the matrices $\underline{A}, \overline{A}$ are called its bounds. Let us define the *center matrix* of $\mathbf{A}$ as $A_c = (1/2)(\underline{A} + \overline{A})$ and the *radius matrix* as $\Delta_A = (1/2)(\overline{A} - \underline{A})$. Then, $\mathbf{A} = [\underline{A}, \overline{A}] = [A_c - \Delta_A, A_c + \Delta_A]$. An *interval vector* is a one-column interval matrix $\mathbf{d} = [\underline{d}, \overline{d}] = \{d \in \mathbb{R}^m : \underline{d} \le d \le \overline{d}\}$, where $\underline{d}, \overline{d} \in \mathbb{R}^m$ and $\underline{d} \le \overline{d}$.

Let $\mathbf{A}$ be an $m \times n$ interval matrix and b be a vector of size m. The following system of interval linear inequalities

$$\mathbf{A}x \le b \tag{3}$$

denotes an *interval linear system*, that is, the *family* of all systems of linear inequalities $Ax \le b$ such that $A \in \mathbf{A}$.

Definition 2 (weak solution). A vector $x \in \mathbb{R}^n$ is called a *weak solution* of the interval linear system $\mathbf{A}x \le b$, if it satisfies $Ax \le b$ for some $A \in \mathbf{A}$. Furthermore, the set

$$\Sigma_{\exists}(\mathbf{A}, b) = \left\{x \in \mathbb{R}^n : \exists A \in \mathbf{A}, Ax \le b\right\} \tag{4}$$

is said to be the *weak solution set* of the system $\mathbf{A}x \le b$.

The weak solution set of an interval linear system is characterized by the following theorem [9].

Theorem 3. *A vector $x \in \mathbb{R}^n$ is a weak solution of $\mathbf{A}x \le b$ if and only if it satisfies $A_c x - \Delta_A |x| \le b$.*

Let $\mathbf{A} \in \mathbb{IR}^{m\times n}$ be an $m \times n$ interval matrix, $b \in \mathbb{R}^m$ be an m-dimensional vector, and $\mathbf{c} \in \mathbb{IR}^n$ be an n-dimensional interval vector. The *family* of linear programming (LP) problems

$$f(A, b, c) = \max\left\{c^T x : Ax \le b\right\} \tag{5}$$

with data satisfying

$$A \in \mathbf{A}, \quad c \in \mathbf{c} \tag{6}$$

is called an interval linear programming (ILP) problem.

In this paper, we are only interested in computing the upper bound $\overline{f}(\mathbf{A}, b, \mathbf{c}) = \sup\{f(A, b, c) : A \in \mathbf{A}, c \in \mathbf{c}\}$. In general, according to Theorem 3, to compute the exact $\overline{f}(\mathbf{A}, b, \mathbf{c})$, in the worst case up to 2^n LP problems have to be solved, one for each orthant. Recall that a (closed) orthant is one of the 2^n subsets of an n-dimensional Euclidean space defined by constraining each Cartesian coordinate axis to be either nonnegative or nonpositive. In each orthant, we consider the following LP problem:

$$\begin{aligned} \max \quad & \sum_{j=1}^{n} c'_j x_j \\ \text{s.t.} \quad & \bigwedge_{0\le i\le m} \sum_{j=1}^{n} A'_{ij} x_j \le b_i, \end{aligned} \tag{7}$$

where

$$c'_j = \begin{cases} \overline{c}_j & \text{if } x_j \geq 0 \\ \underline{c}_j & \text{if } x_j < 0 \end{cases}$$
$$A'_{ij} = \begin{cases} \underline{A}_{ij} & \text{if } x_j \geq 0 \\ \overline{A}_{ij} & \text{if } x_j < 0. \end{cases} \tag{8}$$

And $\overline{f}(\mathbf{A}, b, \mathbf{c})$ will be the the maximum over all the optimal values of the 2^n LP problems with one per each orthant.

3. Robustness Analysis via Self-Composition

3.1. Robustness of Programs. In this paper, we follow the definition for robustness of programs used by Majumdar and Saha [5]. Let f be a function with inputs $x_1, \ldots, x_n$ and output y; that is, $y = f(x_1, \ldots, x_n)$. The function f is said to be (δ, ϵ)-*robust* in the ith input x_i if a perturbation of at most δ in the input x_i can only cause a change of at most ϵ in the output; that is,

$$\forall x_i, x'_i \cdot \left|x_i - x'_i\right| \leq \delta \Longrightarrow \left|f(x_1, \ldots, x_i, \ldots, x_n) - f(x_1, \ldots, x'_i, \ldots, x_n)\right| \leq \epsilon, \tag{9}$$

where $\delta, \epsilon \in \mathbb{R}$ are nonnegative constant parameters specified by users. Recall that we consider the perturbation over only one input at a time while assume that there is no perturbation over all other inputs at the same time.

Moreover, in practice, users may be interested in the maximum output change of y with respect to x_i and δ; that is,

$$\overline{\epsilon}_\delta \overset{\text{def}}{=} \max_{x,x'} \left\{ \left|y - y'\right| \;\middle|\; \begin{array}{c} y = f(x_1, \ldots, x_i, \ldots, x_n) \\ y' = f(x_1, \ldots, x'_i, \ldots, x_n) \\ \left|x_i - x'_i\right| \leq \delta \end{array} \right\}. \tag{10}$$

Similarly, users may be interested in the maximum input perturbation allowed over x_i with respect to y and ϵ; that is,

$$\overline{\delta}_\epsilon \overset{\text{def}}{=} \max_{y,y'} \left\{ \left|x_i - x'_i\right| \;\middle|\; \begin{array}{c} y = f(x_1, \ldots, x_i, \ldots, x_n) \\ y' = f(x_1, \ldots, x'_i, \ldots, x_n) \\ \left|y - y'\right| \leq \epsilon \end{array} \right\}. \tag{11}$$

Example 4. Consider the program shown in Figure 2, which implements a piece-wise linear function. When $x = 1.001$, the two branches give the same result $y = 1002.001$ in exact real arithmetic (assuming floats are reals). It is easy to see that in exact real arithmetic, this program is $(0.1, \epsilon_0)$-*robust* for all $\epsilon_0 \geq 100.1$ but is not $(0.1, \epsilon_1)$-*robust* for all $\epsilon_1 < 100.1$. This can be deduced by observing that in exact real arithmetic, given the input perturbation $\delta = 0.1$, the maximum output change of y is $\overline{\epsilon}_\delta = 100.1$; given the output change $\epsilon = 100.1$, the maximum input perturbation allowed over x is $\overline{\delta}_\epsilon = 0.1$.

```
(1) float piecewise_linear(float x) {
(2)     float y;
(3)     if(x < 1.001)
(4)         y = x + 1001.0;
(5)     else
(6)         y = x * 1001.0;
(7)     return y;
(8) }
```

FIGURE 2: A floating-point program `piecewise_linear`.

3.2. Self-Composition. The idea of self-composition is firstly used in the field of secure information flow [10, 11] to characterize noninterference. Let P be a program and P' be a copy of P with each variable x in P replaced by a fresh variable x'. Using Hoare triples, noninterference can be characterized as

$$\left\{L = L'\right\} \;P; P'\; \left\{L = L'\right\}, \tag{12}$$

where L denotes low-security variables. In other words, it requires that running two instances of the same program with equal low-security values and arbitrary high-security values results in equal low-security values. Hence, via self-composition, a secure information flow property of P reduces to a reachability property over single program executions of the program $P; P'$.

In this paper, we would like to leverage the idea of self-composition to reduce the robustness problem of a program P into an equivalent reachability problem over $P; P'$. Assume that program P has n input variables $x_1, \ldots, x_n$ and an output variable y. Similarly, using Hoare triples, the (δ, ϵ)-*robustness* of program P over the ith input x_i can be characterized as

$$\left\{ \left|x_i - x'_i\right| \leq \delta \wedge \bigwedge_{1 \leq j \leq n, j \neq i} x_j = x'_j \right\} \;P; P'\; \left\{ \left|y - y'\right| \leq \epsilon \right\}. \tag{13}$$

Example 5. Consider again the program `piecewise_linear` in Figure 2. The self-composition of the function body is shown in Figure 3. To express the robustness property, we add the assumption $|x - x'| \leq \delta$ as a precondition at the beginning of the self-composed program and add an assertion $|y - y'| \leq \epsilon$ as a postcondition at the end.

Essentially, the copied program P' has the same program code as P but uses variables with different initial values. Hence, there exists inherent symmetry and redundancy in the self-composed programs. In order to make the following analysis and verification process for self-composed programs easier, program transformations can be used to optimize the self-composed programs. In the field of secure information flow analysis, Terauchi and Aiken [12] proposed type-directed transformation to improve self-composition. The main idea of type-directed transformation is not to self-compose branch (or loop) statements when the branch (or loop) condition is only dependent on the values of low-security variables. In addition, for an assignment statement

```
(1) assume(−δ ≤ x − x′ ≤ δ)
(2)
(3) if (x < 1.001)
(4)     y = x + 1001.0;
(5) else
(6)     y = x ∗ 1001.0;
(7)
(8) if (x′ < 1.001)
(9)     y′ = x′ + 1001.0;
(10) else
(11)    y′ = x′ ∗ 1001.0;
(12)
(13) assert(−ϵ ≤ y − y′ ≤ ϵ)
```

FIGURE 3: Self-composition of the `piecewise_linear` program for robustness analysis.

```
(1) float min_plus1(float x, float y){
(2)     float z;
(3)     int i;
(4)     i = 0;
(5)     while (i < 10) {
(6)         x = x + 0.1;
(7)         i = i + 1;
(8)     }
(9)     if (y <= x) z = y;
(10)    else z = x;
(11)    return z;
(12) }
```

FIGURE 4: A floating-point program `min_plus1`.

```
(1) assume(−δ ≤ x − x′ ≤ δ and y = y′)
(2)
(3) i = 0; i′ = i;
(4) while (i < 10) {
(5)     x = x + 0.1; x′ = x′ + 0.1;
(6)     i = i + 1; i′ = i;
(7) }
(8) if (y <= x) z = y;
(9) else z = x;
(10) if (y′ <= x′) z′ = y′;
(11) else z′ = x′;
(12)
(13) assert(−ϵ ≤ z − z′ ≤ ϵ)
```

FIGURE 5: Transformed self-composition of the `min_plus1` program for robustness analysis.

$\{x := e;\}$, when the right-hand expression e is only dependent on the values of low-security variables, its self-composition is simplified as $\{x := e; x' := x;\}$.

With respect to robustness, a similar transformation can be applied. Intuitively, we could consider the perturbed input variable x_i as a high-security variable and all other input variables x_j's as low-security variables where $j \neq i$. Hence, similarly to type directed transformation, we do not self-compose branch (or loop) statements when the branch (or loop) condition is not dependent on the values of perturbed input variables. For an assignment statement $\{x := e;\}$, when the right-hand expression e is not dependent on the values of perturbed input variables, its self-composition is simplified as $\{x := e; x' := x;\}$.

Example 6. Consider the function *min_plus*1 shown in Figure 4, which implements $\min(x + 1, y)$ by adding 0.1 to x ten times. The optimized self-composition result of the function body after applying transformation is given in Figure 5, when we consider the perturbation over the input variable x (while assuming no perturbation over y). More specifically, since the loop condition $i < 10$ in the original program is not dependent on the value of the perturbed input variable x, we do not self-compose the loop statement and thus there is only one loop in the transformed resulting self-composed program.

3.3. Robustness Analysis of Self-Composed Programs. Via self-composition, the robustness analysis problem can be reduced to solving a standard reachability (safety) problem. The recent success of automatic analysis and verification tools (such as SLAM [13], CBMC [14], and ASTRÉE [15]) aiming at checking reachability properties in programs makes this approach promising. In the following, we will present two popular reachability analysis approaches that fit for analyzing robustness, that is, software model checking and symbolic execution.

3.3.1. Checking Robustness by Software Model Checking. Software model checking [16] provides an automatic approach to check whether a program satisfies a property by exploring the state space of the program. For the robustness analysis problem, the property to be checked is an assertion at the end of the self-composed programs stating that the output change is bounded by ϵ, that is, `assert` $(-\epsilon \leq y - y' \leq \epsilon)$. A main advantage of using software model checking is that it will generate a counterexample when the robustness property does not hold. The counterexample shows an execution trace which violates the robustness property. A counterexample is very helpful for the users to identify the source of nonrobustness.

3.3.2. Finding Maximum Output Change (or Input Perturbation) by Symbolic Execution. Symbolic execution [17, 18] is a technique to analyze a program by executing the program with symbolic rather than concrete values as program inputs. The process of symbolic execution essentially generates and explores a symbolic execution tree which represents all execution paths followed during the process. Each tree node represents a symbolic execution state, while each edge represents a program transition between the states. At any tree node, the symbolic execution state includes a program counter, a *path condition* (PC) that encodes the constraints on the symbolic inputs to reach that node, a *path function* (PF) that represents the current values of the program variables as function of symbolic inputs when the path condition holds true. The path condition is a boolean expression over the symbolic inputs. The path function describes the expected result of the program, under the given path condition. Due to conditional branches and loops in a program, the symbolic execution of a program will result in a set of paths, each of which is described by a pair $\langle \mathrm{PC}, \mathrm{PF} \rangle$ of the path condition PC and the associated path function PF.

We now show how to use symbolic execution to conduct a robustness analysis of program P with inputs $x_1, \ldots, x_n$ and output y. First, the analysis algorithm performs symbolic execution on the self-composed program $P; P'$. Assume that the algorithm collects, at the end of the self-composed program, a set $\mathcal{S}$ of pairs $\langle \mathrm{PC}, \mathrm{PF} \rangle$ of the path condition PC and the associated path function PF. Then for each $s \triangleq \langle \mathrm{PC}, \mathrm{PF} \rangle \in \mathcal{S}$, we compute the maximum output change $\overline{\epsilon}_{\delta}^{s}$:

$$\begin{aligned} \max \quad & \left| \mathrm{PF}_y - \mathrm{PF}_{y'} \right| \\ \text{s.t.} \quad & \left(\left| x_i - x_i' \right| \le \delta \wedge \bigwedge_{1 \le j \le n, j \ne i} x_j = x_j' \right) \wedge \mathrm{PC}. \end{aligned} \tag{14}$$

Here, PF_y and $\mathrm{PF}_{y'}$ denote the symbolic expressions that the path function PF maps the variables y and y' to, respectively. Let $\overline{\epsilon}_{\delta}$ be the maximum element of $\{\overline{\epsilon}_{\delta}^{s} \mid s \in \mathcal{S}\}$; that is, $\overline{\epsilon}_{\delta} = \max(\{\overline{\epsilon}_{\delta}^{s} \mid s \in \mathcal{S}\})$. If $\overline{\epsilon}_{\delta} \le \epsilon$, then the original program P is (δ, ϵ)-*robust*.

Similarly, given the bound of output change ϵ, computing the maximum allowed input perturbation is reduced to solving a series of the following optimization problems for each $s \triangleq \langle \mathrm{PC}, \mathrm{PF} \rangle \in \mathcal{S}$:

$$\begin{aligned} \max \quad & \left| x_i - x_i' \right| \\ \text{s.t.} \quad & \left(\bigwedge_{1 \le j \le n, j \ne i} x_j = x_j' \right) \wedge \mathrm{PC} \wedge \left| \mathrm{PF}_y - \mathrm{PF}_{y'} \right| \le \epsilon. \end{aligned} \tag{15}$$

And $\overline{\delta}_{\epsilon}$ will be the maximum element of $\{\overline{\delta}_{\epsilon}^{s} \mid s \in \mathcal{S}\}$; that is, $\overline{\delta}_{\epsilon} = \max(\{\overline{\delta}_{\epsilon}^{s} \mid s \in \mathcal{S}\})$.

4. Robustness Analysis of Floating-Point Programs

In this section, we consider the robustness analysis problem of floating-point programs. In Section 3.3, we propose to utilize software model checking and symbolic execution to perform robustness analysis of self-composed programs (in exact real arithmetic). However, most existing software model checkers and symbolic execution tools can not be directly applied to floating-point programs, since they rely on constraint solvers that often assume good algebraic properties such as associativity and distributivity over the reals which do not hold for floating-point arithmetic. To handle floating-point arithmetic, we have to resort to bit-precise modeling of floating-point arithmetic or abstracting floating-point arithmetic to real number arithmetic.

CBMC (C Bounded Model Checker) [14] is one of the few software model checkers that have considered floating-point arithmetic. CBMC employs a sound and complete decision procedure for floating-point arithmetic [19, 20]. It precisely encodes floating-point operations as functions on bit-vectors. Each floating-point operation is further modeled as a formula in propositional logic. The formula is then handled by a SAT-solver in the backend to check for satisfiability.

When we consider symbolic execution of floating-point programs, both the path condition and the path function will involve floating-point expressions. Hence, to compute the maximum output change (or maximum allowed input perturbation), we need optimization methods supporting floating-point constraints. However, as far as we know, even for linear programming, there is no available sound solver supporting floating-point constraints. To this end, in this paper, we abstract the optimization problem with floating-point constraints into an interval linear programming problem (i.e., linear programming problem with interval coefficients) over the reals. The main idea is to use the so-called *floating-point linearization* technique [21, 22] to abstract floating-point expressions into linear real number expressions with interval coefficients (in the form of $\Sigma_i [a_i, b_i] x_i$).

4.1. Floating-Point Abstraction.

In this subsection, we will explain how to abstract floating-point expressions into interval linear expressions over the reals.

First, let us consider the upper bound on rounding errors due to one floating-point operation. Let $R_{\mathbf{f},r}(x)$ denote the floating-point rounding function that maps a real number x to a floating-point number (or a runtime error due to, for example, overflows) with respect to the floating-point format $\mathbf{f}$ and the rounding mode r. The amount of the rounding error due to $R_{\mathbf{f},r}(x)$ depends on the category of x.

(i) If x is in the range of normalized numbers, then $|R_{\mathbf{f},r}(x) - x| \le \varepsilon_{\mathrm{rel}} \cdot |x|$ where $\varepsilon_{\mathrm{rel}} = 2^{-\mathbf{p}}$ (wherein $\mathbf{p}$ is the number of bits of fraction in the significand of the floating-point format $\mathbf{f}$). In this case we consider the relative rounding error $\varepsilon_{\mathrm{rel}}$.

(ii) If x is in the range of denormalized number, then $|R_{\mathbf{f},r}(x) - x| \le \varepsilon_{\mathrm{abs}}$, where $\varepsilon_{\mathrm{abs}} = mf_{\mathbf{f}}$ (wherein $mf_{\mathbf{f}}$ is the smallest nonzero positive denormalized floating-point number in the floating-point format $\mathbf{f}$, which is also the gap between two neighboring denormalized numbers). In this case, we consider the absolute rounding error $\varepsilon_{\mathrm{abs}}$.

The rounding errors of these two cases can be unified as

$$\left| R_{\mathbf{f},r}(x) - x \right| \le \max\left(\varepsilon_{\mathrm{rel}} \cdot |x|, \varepsilon_{\mathrm{abs}} \right). \tag{16}$$

Since max is not a linear operation, we derive an overapproximation

$$\left| R_{\mathbf{f},r}(x) - x \right| \le \varepsilon_{\mathrm{rel}} \cdot |x| + \varepsilon_{\mathrm{abs}}. \tag{17}$$

Furthermore, when $b \ge 0$, $|y| \le b$ is equivalent to $y = [-1, 1] \times b$. Hence,

$$R_{\mathbf{f},r}(x) - x = [-1, 1] \left(\varepsilon_{\mathrm{rel}} \cdot |x| + \varepsilon_{\mathrm{abs}} \right); \tag{18}$$

that is,

$$R_{\mathbf{f},r}(x) = [1-\varepsilon_{\text{rel}}, 1+\varepsilon_{\text{rel}}] \times x + [-\varepsilon_{\text{abs}}, \varepsilon_{\text{abs}}]. \tag{19}$$

In general, we could abstract floating-point operations into interval linear expressions in real number semantics. For example,

$$x \oplus_{\mathbf{f},r} y, \tag{20}$$

that is,

$$R_{\mathbf{f},r}(x+y) \tag{21}$$

can be abstracted into

$$[1-\varepsilon_{\text{rel}}, 1+\varepsilon_{\text{rel}}] \times (x+y) + [-\varepsilon_{\text{abs}}, \varepsilon_{\text{abs}}]; \tag{22}$$

that is,

$$[1-\varepsilon_{\text{rel}}, 1+\varepsilon_{\text{rel}}] \times x + [1-\varepsilon_{\text{rel}}, 1+\varepsilon_{\text{rel}}] \times y + [-\varepsilon_{\text{abs}}, \varepsilon_{\text{abs}}]. \tag{23}$$

The advantage of this kind of rounding mode insensitive floating-point abstractions is that the result is sound with respect to arbitrary rounding modes, since $R_{\mathbf{f},r}(x)$ always satisfies $R_{\mathbf{f},-\infty}(x) \le R_{\mathbf{f},r}(x) \le R_{\mathbf{f},+\infty}(x)$ while $|R_{\mathbf{f},r}(x) - x| \le \varepsilon_{\text{rel}} \cdot |x| + \varepsilon_{\text{abs}}$ has already taken into account the extreme cases of $r = -\infty$ and $r = +\infty$. This is of practical importance, since we may not know the exact rounding mode for each floating-point operation. For example, C99 provides the `fesetround()` function to set the current rounding mode. Of course, when we know the exact rounding mode for the floating-point operation, we could make the floating-point abstraction more precise. For example, if the current rounding mode is toward nearest, then

$$R_{\mathbf{f},n}(x) = \left[1-\frac{\varepsilon_{\text{rel}}}{2}, 1+\frac{\varepsilon_{\text{rel}}}{2}\right] \times x + \left[-\frac{\varepsilon_{\text{abs}}}{2}, \frac{\varepsilon_{\text{abs}}}{2}\right]. \tag{24}$$

In addition, if we know the range of x, we may also define more precise floating-point abstractions. For example, if we know that x is in the range of denormalized numbers, then

$$R_{\mathbf{f},r}(x) = x + [-\varepsilon_{\text{abs}}, \varepsilon_{\text{abs}}]. \tag{25}$$

For the sake of generality, in this paper, we use the following rounding mode insensitive floating-point abstraction:

$$R^{\#}_{\mathbf{f},?}(x) = [1-\varepsilon_{\text{rel}}, 1+\varepsilon_{\text{rel}}] \times x + [-\varepsilon_{\text{abs}}, \varepsilon_{\text{abs}}], \tag{26}$$

where we assume $|x| < Mf_{\mathbf{f}}$.

More clearly, we use the following abstraction for floating-point arithmetic:

$$\begin{aligned}
R^{\#}_{\mathbf{f},?}(x \oplus_{\mathbf{f},?} y) &= [1-\varepsilon_{\text{rel}}, 1+\varepsilon_{\text{rel}}] \times x + [1-\varepsilon_{\text{rel}}, 1+\varepsilon_{\text{rel}}] \\
&\quad \times y + [-\varepsilon_{\text{abs}}, \varepsilon_{\text{abs}}] \\
R^{\#}_{\mathbf{f},?}(x \ominus_{\mathbf{f},?} y) &= [1-\varepsilon_{\text{rel}}, 1+\varepsilon_{\text{rel}}] \times x + [-1-\varepsilon_{\text{rel}}, -1+\varepsilon_{\text{rel}}] \\
&\quad \times y + [-\varepsilon_{\text{abs}}, \varepsilon_{\text{abs}}] \\
R^{\#}_{\mathbf{f},?}(x \otimes_{\mathbf{f},?} y) &= [1-\varepsilon_{\text{rel}}, 1+\varepsilon_{\text{rel}}] \times x \times y + [-\varepsilon_{\text{abs}}, \varepsilon_{\text{abs}}] \\
R^{\#}_{\mathbf{f},?}(x \oslash_{\mathbf{f},?} y) &= [1-\varepsilon_{\text{rel}}, 1+\varepsilon_{\text{rel}}] \times \frac{x}{y} + [-\varepsilon_{\text{abs}}, \varepsilon_{\text{abs}}].
\end{aligned} \tag{27}$$

Specially, for a constant number c that appears in the source code, we use the following abstraction:

$$R^{\#}_{\mathbf{f},?}(c) = \left[R^{\#}_{\mathbf{f},-\infty}(c), R^{\#}_{\mathbf{f},+\infty}(c)\right]. \tag{28}$$

4.2. Symbolic Execution of Abstracted Floating-Point Programs. From Section 4.1, we see that floating-point expressions can be soundly abstracted into real number expressions with interval coefficients. Since the multiplication $x \times y$ and division x/y are not linear expressions when both x and y are not constant numbers, in order to obtain linear expressions with interval coefficients, we replace y with its interval range denoted as $[\underline{y}, \overline{y}]$. In symbolic execution, y is always an expression over the symbolic input values. We assume users provide the interval ranges for those symbolic input values. Then, all floating-point expressions can be abstracted as interval linear expressions. Therefore, the resulting path conditions of symbolic execution consist of interval linear constraints while the resulting path functions consist of interval linear expressions.

Finally, the problems of computing the maximum output change and the maximum allowed input perturbation are reduced to solving a series of interval linear programming problems. For example, computing the maximum output change requires the solutions of the following interval linear programming problems:

$$\begin{aligned}
\max \quad & \Sigma_i [\underline{a}_i, \overline{a}_i] \times x_i + \Sigma_i [\underline{a}'_i, \overline{a}'_i] \times x'_i + b \\
\text{s.t.} \quad & \left(|x_i - x'_i| \le \delta \wedge \bigwedge_{1\le j\le n, j\ne i} x_j = x'_j \right) \\
& \wedge \bigwedge_k \Sigma_i [\underline{A}_{ki}, \overline{A}_{ki}] \times x_i + \Sigma_i [\underline{A}'_{ki}, \overline{A}'_{ki}] \times x'_i \le c_k,
\end{aligned} \tag{29}$$

where $\Sigma_i[\underline{a}_i, \overline{a}_i] \times x_i + \Sigma_i[\underline{a}'_i, \overline{a}'_i] \times x'_i + b$ denotes the abstracted output change $\text{PF}_y - \text{PF}_{y'}$ (or $\text{PF}_{y'} - \text{PF}_y$) while $\bigwedge_k \Sigma_i[\underline{A}_{ki}, \overline{A}_{ki}] \times x_i + \Sigma_i[\underline{A}'_{ki}, \overline{A}'_{ki}] \times x'_i \le c_k$ denotes the abstracted PC.

Example 7. Consider the self-composed program `piecewise_linear` in Example 5. Suppose we would like to compute the maximum output change, given the input perturbation $\delta = 0.1$ over x. The self-composed program includes four paths overall. Let us consider for example the path that takes the `else` branch in both the unprimed program P and the primed program P'. Since x, y are of `float` type, $\varepsilon_{\text{rel}} = 2^{-23}$ and $\varepsilon_{\text{abs}} = 2^{-149}$ for the 32-bit single precision floating-point format. We will have

$$\begin{aligned}
\text{PC} &: x \ge R^{\#}_{\mathbf{f},-\infty}(1.001) \wedge x' \ge R^{\#}_{\mathbf{f},-\infty}(1.001) \\
\text{PF}_y &: \left[1-2^{-23}, 1+2^{-23}\right] \\
&\quad \times \left[R^{\#}_{\mathbf{f},-\infty}(1001.0), R^{\#}_{\mathbf{f},+\infty}(1001.0)\right] \times x \\
&\quad + \left[-2^{-149}, 2^{-149}\right]
\end{aligned}$$

$$\begin{aligned}\mathrm{PF}_{y'}: &\left[1-2^{-23}, 1+2^{-23}\right] \\ &\times\left[R^{\#}_{\mathbf{f},-\infty}(1001.0), R^{\#}_{\mathbf{f},+\infty}(1001.0)\right]\times x' \\ &+\left[-2^{-149}, 2^{-149}\right].\end{aligned} \tag{30}$$

Thus, we get the following interval linear programming problem:

$$\begin{aligned}\max\quad &\left[1-2^{-23}, 1+2^{-23}\right] \\ &\times\left[R^{\#}_{\mathbf{f},-\infty}(1001.0), R^{\#}_{\mathbf{f},+\infty}(1001.0)\right]\times x \\ &+\left[-1-2^{-23}, -1+2^{-23}\right] \\ &\times\left[R^{\#}_{\mathbf{f},-\infty}(1001.0), R^{\#}_{\mathbf{f},+\infty}(1001.0)\right]\times x' \\ &+\left[-2^{-148}, 2^{-148}\right]\end{aligned} \tag{31}$$

$$\begin{aligned}\text{s.t.}\quad & x-x'\le 0.1 \wedge -x+x'\le 0.1 \\ &\wedge -x\le -R^{\#}_{\mathbf{f},-\infty}(1.001)\wedge -x'\le -R^{\#}_{\mathbf{f},-\infty}(1.001).\end{aligned} \tag{32}$$

Solving the above interval linear programming problem by the method described in Section 2.2 will give us 100.10023889571252. After we deal with all other paths in the same way, we will find that 100.10023889571252 is the maximum output change with respect to the given input perturbation 0.1. Hence, the program `piecewise_linear` in floating-point arithmetic is at least (0.1, 100.10023889571252)-*robust*.

5. Implementation and Experimental Results

We have implemented a robustness analysis tool RAFP, based on the symbolic execution and floating-point abstraction techniques presented in Section 4. Given an input perturbation δ over one input variable of the program, RAFP can compute the maximum output change. Furthermore, if the user also provides a candidate output change ϵ and would like to check whether the program is (δ, ϵ)-*robust*, RAFP will check this property during the process of computing maximum output change and will stop once one path violating the property is found. Also, given an output change ϵ, RAFP can compute the maximum allowed input perturbation for floating-point programs. RAFP is built on top of Symbolic PathFinder (SPF) [23] which is a symbolic execution engine for Java programs. We use SPF to extract the path conditions together with the associated path functions. For linear programming, RAFP makes use of the Java Binding for GLPK (GNU Linear programming kit) called GLPK-Java [24].

To conduct experiments on checking robustness properties of floating-point programs via software model checking, we choose CBMC (C Bounded Model Checker) [14] which implements bounded model checking for ANSI-C programs using SAT/SMT solvers. CBMC utilizes a bit-precise modeling for floating-point operations and employs a sound and complete decision procedure for floating-point arithmetic. CBMC provides an option `--floatbv` to use IEEE floating point arithmetic and options for choosing rounding modes. However, CBMC does not support to use different rounding modes for the floating-point operations in the same program. In other words, all floating-point operations in a program are of the same rounding mode during the analysis. We use the default rounding mode `--round-to-nearest` during our experiments. Moreover, CBMC provides `__CPROVER_assume()` and `__CPROVER_assert()` statements, which are needed for robustness analysis of self-composed programs. Both statements take Boolean conditions. The `__CPROVER_assume()` statement restricts that the program traces should satisfy the assumed condition. For the `__CPROVER_assert()` statement, CBMC will check whether the asserted condition holds true for all runs of the program.

We have conducted experiments on a selection of benchmark examples using both RAFP and CBMC. Table 1 shows the comparison of performance and the resulting output changes. The column "δ_{in}" shows the considered input perturbation over one input variable of the program. The column "$\epsilon_{\max}$" shows the resulting maximum output change computed by RAFP with respect to the given input perturbation. The column "ϵ_{unr}" gives the largest possible output change that we have tried with CBMC such that the program is not $(\delta_{\text{in}}, \epsilon_{\text{unr}})$-*robust* with respect to the given input perturbation. The column "ϵ_r" gives the smallest output change that we have tried with CBMC such that the program is $(\delta_{\text{in}}, \epsilon_r)$-*robust* with respect to the given input perturbation (Note that CBMC can be used only to check whether a program is (δ, ϵ)-*robust* and can not be used to compute the amount of output change with respect to the given input perturbation. During our experiments, we try CBMC with different candidate values of ϵ to find ϵ_{unr} and ϵ_r.). Since CBMC uses the same rounding mode for all floating-point operations in the same program during the analysis, the output change is always 0 when the given input perturbation is 0. Hence, for those rows that specify input perturbation as 0, we do not need to run CBMC and thus we mark the table entry with $\star$ in this case. Our tool RAFP utilizes rounding mode insensitive floating-point abstraction and thus in principle it holds that $\epsilon_{\max} \ge \epsilon_r \ge \epsilon_{\text{unr}}$, which is confirmed by the experimental results.

The program `piecewise_linear` corresponds to the program shown in Example 4. `Max1`, `MorePaths` come from JPF Continuity [25]. `Max1` is a floating-point program that implements $\max(x, y)$, and thus it is $(\delta_{\text{in}}, \delta_{\text{in}})$-*robust*. `MorePaths` is a floating-point program that involves both a step function and a max function, and thus it is $(\delta_{\text{in}}, 1.0)$-*robust* for all $\delta_{\text{in}} \le 1.0$. `Orientation` (which corresponds to the program shown in Figure 1) together with `Filtered_Orientation` are extracted from the computational geometry algorithms library CGAL [26] and address robust geometric computation. `Filtered_Orientation` is an improved version of `Orientation` via static filter technique. The approximate result of computing the sign of a determinant

Table 1: Experimental results for benchmark examples.

Program	δ_{in}	RAFP		CBMC			
		ϵ_{max}	t (ms)	ϵ_{unr}	t (ms)	ϵ_r	t (ms)
`piecewise_linear`	0	$2.3889571220452006e-4$	29	$\star$	$\star$	$\star$	$\star$
	0.01	10.010238895712442	33	?	>1 h	?	>1 h
	0.1	100.10023889571252	34	?	>1 h	?	>1 h
`Max1`	0	$1.4012987984203268e-45$	44	$\star$	$\star$	$\star$	$\star$
	0.01	0.010000000000000007	55	0.0099	215	0.01	16066
	0.1	0.10000000000000006	54	0.099	227	0.1	16262
`MorePaths`	0	1.0000000027939686	59	$\star$	$\star$	$\star$	$\star$
	0.01	1.0000000027939686	79	0.99	279	1.0	379
	0.1	1.0000000027939686	86	0.99	314	1.0	471
`Orientation`	0	2.0	55	$\star$	$\star$	$\star$	$\star$
	$0.1e-8*\mathscr{E}$	2.0	68	0	1840	1.0	6845
	$0.1e-3*\mathscr{E}$	2.0	73	0	1338	1.0	13327
	$0.1e-2*\mathscr{E}$	2.0	80	1.0	14165	2.0	413
`Filtered_Orientation`	0	1.0	54	$\star$	$\star$	$\star$	$\star$
	$0.1e-8*\mathscr{E}$	1.0	70	0	1044	1.0	29641
	$0.1e-2*\mathscr{E}$	1.0	73	0	898	1.0	30261
	$0.1*\mathscr{E}$	2.0	75	0	719	1.0	26463
	$\mathscr{E}$	2.0	68	1.0	13206	2.0	654

is compared with a given positive filter bound $\mathscr{E}$ (rather than compared with zero). When the approximate result is in the interval $[-\mathscr{E}, \mathscr{E}]$, `Filtered_Orientation` gives 0. During our experiments, we set $\mathscr{E} = 1.5e-5$ (and for the sake of comparison, we express the input perturbation in terms of $\mathscr{E}$ also for `Orientation` although here $\mathscr{E}$ does not appear). The outputs of `Orientation` and `Filtered_Orientation` are always −1 (negative), 0 (zero), or 1 (positive). Hence, in Table 1, the resulting output changes for these two programs are always 0, 1.0, or 2.0. From Table 1, we could find that `Filtered_Orientation` is more robust than `Orientation`. For example, given the input perturbation $\delta = 0.1e-2*\mathscr{E}$, CBMC finds that for $\epsilon = 1.0$, `Filtered_Orientation` is robust while `Orientation` is not. Similarly, given the input perturbation $\delta = 0.1e-2*\mathscr{E}$, RAFP gives $\epsilon_{max} = 1.0$ for `Filtered_Orientation` but gives $\epsilon_{max} = 2.0$ for `Orientation`.

The column "t (ms)" presents the analysis times in milliseconds when the analyzers run on a 2.5 GHz PC with 4 GB of RAM running Windows 7. (RAFP runs further on a Java Virtual Machine (JVM) while CBMC runs further on a virtual machine VMWare running Fedora 12.) From Table 1, we could see that RAFP outperforms CBMC in time efficiency. Especially for `piecewise_linear`, CBMC could not even finish the analysis process in 1 hour. The low efficiency of CBMC is because that CBMC uses a sound and complete decision procedure for floating-point arithmetic. Especially, the multiplication and division floating-point operations may generate formulae that are expensive to decide and quite hard for SAT solvers to solve [27]. Hence, checking robustness properties of floating-point programs via CBMC may have limitations in scalability due to the current expensive decision procedures for floating-point logic. During our experiments, the approach via symbolic execution of abstracted floating-point programs is much more efficient. In principle, symbolic execution may suffer from the path explosion problem. However, the recent success of symbolic execution tools such as KLEE [28] on analyzing large-scale programs [17] makes this approach promising.

6. Related Work

6.1. Robustness Analysis of Programs. Robustness is a standard correctness property for control systems [1]. Robustness analysis of programs has received increasing attention in the recent years. Majumdar and Saha [5] took a first step toward analyzing the robustness of programs in control systems. They also utilized symbolic execution and optimization techniques to compute the maximum difference in program outputs with respect to the given input perturbation. However, they assumed exact real arithmetic in the program. Continuity as one aspect of robustness for software was firstly considered in [29]. Recently, Chaudhuri et al. presented logic-based mostly automated methods to determine whether a program is continuous [2] or Lipschitz continuous [3, 4], and more recently to determine whether a decision-making program is consistent under uncertainty [30]. Quite recently, Shahrokni and Feldt [31] conducted a systematic review of software robustness. However, much existing work on robustness analysis does not handle floating-point arithmetic in the program. Bushnell [25] presented a symbolic execution based approach to identify continuities and discontinuities associated with path condition boundaries for floating-point software, but it did not consider the true floating-point semantics. Besides, Gazeau et al. [32]

presented a nonlocal method for proving the robustness of floating-point programs but which needs much manual work. Recently, Goubault and Putot [33] proposed an abstract interpretation based robustness analysis method for finite precision implementations.

6.2. Safety Analysis of Floating Point Programs. Monniaux [7] described common pitfalls in analyzing and verifying floating-point programs. Abstract interpretation [34] based techniques have shown quite successful on analysis of floating-point programs. In [35], Goubault analyzed the origin of the loss of precision in floating-point programs based on abstract interpretation. Following this direction, a static analyzer FLUCTUAT [36] was developed. The abstract interpretation based static analyzer ASTRÉE [15] checks for floating-point run-time errors based on the computed set of reachable values for floating-point variables. As in ASTRÉE, we rely on the floating-point abstraction technique of [21] to soundly abstract floating-point expressions into ones over the field of reals. Chen et al. [37, 38] utilized interval linear constraints to design numerical abstract domains and to construct sound floating-point implementations [39]. Ivančić et al. [40] used bounded model checking based on SMT solvers to detect numerical instabilities in floating-point programs, based on a mixed integer-real model for floating-point variables and operations. Brain et al. [41] recently improved the bit-precise decision procedure for the theory of floating-point arithmetic based on a strict lifting of the conflict-driven clause learning algorithm in modern SAT solvers to abstract domains. Barr et al. [42] presented a method to automatically detect the floating-point exception through symbolic execution.

6.3. Self-Composition. The idea of self-composition is firstly used in the field of secure information flow [10, 11], to characterize noninterference. Terauchi and Aiken [12] proposed the type-directed transformation approach to make self-composition work in practice with off-the-shelf automatic safety analysis tools. Recently, Barthe et al. [43] proposed a general notion of product program that is beneficial to relational verification, which could be considered as the generalization of self-composition. Kovacs et al. [44] presented a general method to analyze 2-hypersafety properties by applying abstract interpretation on the self-compositions of the control flow graphs of programs.

7. Conclusion

We have proposed a self-composition based approach for robustness analysis of programs, which enables making use of off-the-shelf automatic reachability analysis tools to analyze robustness properties of programs. Then, we have shown how to use software model checking and symbolic execution techniques on self-composed programs to analyze program robustness properties. In particular, we have considered the robustness analysis problem of floating-point programs. To deal with floating-point arithmetic during symbolic execution, we have utilized a rounding mode insensitive floating-point abstraction to abstract floating-point expressions into interval linear expressions in exact real arithmetic. On this basis, the maximum output change (when given the input perturbation) or maximum allowed input perturbation (when given the input perturbation) are computed based on symbolic execution and interval linear programming for abstracted floating-point programs. Experimental results of our prototype implementation are encouraging.

It remains for future work to exploit the intrinsic symmetry of self-composed programs to reduce the number of considered paths during robustness analysis. We also plan to improve the prototype implementation and to conduct more experiments on larger realistic floating-point programs.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

This work is supported by the 973 Program under Grant no. 2014CB340703, the 863 Program under Grant no. 2011AA010106, the NSFC under Grant nos. 61202120, 61120106006, and 91318301 and the SRFDP under Grant no. 20124307120034.

References

[1] S. Pettersson and B. Lennartson, “Stability and robustness for hybrid systems,” in *Proceedings of the 35th IEEE Conference on Decision and Control*, pp. 1202–1207, December 1996.

[2] S. Chaudhuri, S. Gulwani, and R. Lublinerman, “Continuity analysis of programs,” in *Proceedings of the 37th Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL '10)*, pp. 57–69, ACM, January 2010.

[3] S. Chaudhuri, S. Gulwani, and R. Lublinerman, “Continuity and robustness of programs,” *Communications of the ACM*, vol. 55, no. 8, pp. 107–115, 2012.

[4] S. Chaudhuri, S. Gulwani, R. Lublinerman, and S. NavidPour, “Proving programs robust,” in *Proceedings of the 19th ACM SIGSOFT Symposium on the Foundations of Software Engineering (FSE '11)*, pp. 102–112, ACM, 2011.

[5] R. Majumdar and I. Saha, “Symbolic robustness analysis,” in *Proceedings of the Real-Time Systems Symposium (RTSS '09)*, pp. 355–363, IEEE, December 2009.

[6] L. Kettner, K. Mehlhorn, S. Pion, S. Schirra, and C. K. Yap, “Classroom examples of robustness problems in geometric computations,” in *Proceedings of the European Symposium on Algorithms (ESA '04)*, vol. 3221 of *Lecture Notes in Computer Science*, pp. 702–713, Springer, 2004.

[7] D. Monniaux, “The pitfalls of verifying floating-point computations,” *ACM Transactions on Programming Languages and Systems*, vol. 30, no. 3, article 12, 2008.

[8] IEEE Computer Society, “IEEE standard for binary floating point arithmetic,” Tech. Rep. ANSI/IEEE Std 745-1985, 1985.

[9] J. Rohn, “Solvability of systems of interval linear equations and inequalities,” in *Linear Optimization Problems with Inexact Data*, pp. 35–77, Springer, 2006.

[10] G. Barthe, P. R. D’Argenio, and T. Rezk, “Secure information flow by self-composition,” in *Proceedings of the 17th IEEE Computer Security Foundations Workshop (CSFW ’04)*, pp. 100–114, IEEE, June 2004.

[11] Á. Darvas, R. Hahnle, and D. Sands, “A theorem proving approach to analysis of secure information flow,” in *Proceedings of the 2nd International Conference on Security in Pervasive Computing (SPC ’05)*, vol. 3450 of *Lecture Notes in Computer Science*, pp. 193–209, Springer, 2005.

[12] T. Terauchi and A. Aiken, “Secure information flow as a safety problem,” in *Proceedings of the International Static Analysis Symposium (SAS ’05)*, vol. 3672 of *Lecture Notes in Computer Science*, pp. 352–367, Springer, 2005.

[13] T. Ball and S. K. Rajamani, “The SLAM project: debugging system software via static analysis,” in *Proceedings of the 29th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL ’02)*, pp. 1–3, ACM Press, January 2002.

[14] E. M. Clarke, D. Kroening, and F. Lerda, “A tool for checking ANSI-C programs,” in *Proceedings of the International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS ’04)*, vol. 2988 of *Lecture Notes in Computer Science*, pp. 168–176, Springer, 2004.

[15] B. Blanchet, P. Cousot, R. Cousot et al., “A static analyzer for large safety-critical software,” in *Proceedings of the ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI ’03)*, pp. 196–207, ACM Press, June 2003.

[16] R. Jhala and R. Majumdar, “Software model checking,” *ACM Computing Surveys*, vol. 41, no. 4, article 21, 2009.

[17] C. Cadar and K. Sen, “Symbolic execution for software testing: three decades later,” *Communications of the ACM*, vol. 56, no. 2, pp. 82–90, 2013.

[18] J. C. King, “Symbolic execution and program testing,” *Communications of the ACM*, vol. 19, no. 7, pp. 385–394, 1976.

[19] A. Brillout, D. Kroening, and T. Wahl, “Mixed abstractions for floating-point arithmetic,” in *Proceedings of the 9th International Conference Formal Methods in Computer Aided Design (FMCAD ’09)*, pp. 69–76, IEEE, November 2009.

[20] L. Haller, A. Griggio, M. Brain, and D. Kroening, “Deciding floating-point logic with systematic abstraction,” in *Proceedings of the International Conference Formal Methods in Computer Aided Design (FMCAD ’12)*, pp. 131–140, IEEE, 2012.

[21] A. Miné, “Relational abstract domains for the detection of floating-point run-time errors,” in *Proceedings of the European Symposium on Programming (ESOP ’04)*, vol. 2986 of *Lecture Notes in Computer Science*, pp. 3–17, Springer, 2004.

[22] A. Miné, *Weakly relational numerical abstract domains [Ph.D. thesis]*, Ecole Polytechnique, Palaiseau, France, 2004.

[23] C. S. Pasareanu, W. Visser, D. H. Bushnell, J. Geldenhuys, P. C. Mehlitz, and N. Rungta, “Symbolic pathfinder: integrating symbolic execution with model checking for java bytecode analysis,” *Automated Software Engineering*, vol. 20, no. 3, pp. 391–425, 2013.

[24] H. Schuchardt, “GLPK for Java,” 2014, http://glpk-java.sourceforge.net/.

[25] D. Bushnell, “Continuity analysis for floating point software,” in *Proceedings of the 4th workshop on Numerical Software Verification (NSV ’11)*, 2011.

[26] E. Fogel and M. Teillaud, “The computational geometry algorithms library cgal,” *ACM Communications in Computer Algebra*, vol. 47, no. 3, pp. 85–87, 2013.

[27] D. Kroening, “The CPROVER User Manual,” http://www.cprover.org/cbmc/doc/manual.pdf.

[28] C. Cadar, D. Dunbar, and D. R. Engler, “KLEE: unassisted and automatic generation of high-coverage tests for complex systems programs,” in *Proceedings of the Symposium on Operating Systems Design and Implementation (OSDI ’14)*, pp. 209–224, USENIX Association, 2008.

[29] D. Hamlet, “Continuity in software systems,” in *Proceedings of the ACM SIGSOFT International Symposium on Software Testing and Analysis (ISSTA ’02)*, pp. 196–200, ACM, July 2002.

[30] S. Chaudhuri, A. Farzan, and Z. Kincaid, “Consistency analysis of decision-making programs,” in *Proceedings of the ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL ’14)*, pp. 555–568, Springer, 2014.

[31] A. Shahrokni and R. Feldt, “A systematic review of software robustness,” *Information & Software Technology*, vol. 55, no. 1, pp. 1–17, 2013.

[32] I. Gazeau, D. Miller, and C. Palamidessi, “A non-local method for robustness analysis of floating point programs,” in *Proceedings of the Workshop on Quantitative Aspects of Programming Languages (QAPL ’12)*, vol. 85 of *Electronic Proceedings in Theoretical Computer Science*, pp. 63–76, 2012.

[33] E. Goubault and S. Putot, “Robustness analysis of finite precision implementations,” in *Proceedings of the Asian Symposium on Programming Languages and Systems (APLAS ’13)*, vol. 8301 of *Lecture Notes in Computer Science*, pp. 50–57, Springer, 2013.

[34] P. Cousot and R. Cousot, “Abstract interpretation: a unified lattice model for static analysis of programs by construction or approximation of fixpoints,” in *Proceedings of the ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL ’77)*, pp. 238–252, ACM, 1977.

[35] E. Goubault, “Static analyses of the precision of floating-point operations,” in *Proceedings of the International Static Analysis Symposium (SAS ’01)*, vol. 2126 of *Lecture Notes in Computer Science*, pp. 234–259, Springer, 2001.

[36] E. Goubault, M. Martel, and S. Putot, “Asserting the precision of floating-point computations: a simple abstract interpreter,” in *Proceedings of the European Symposium on Programming (ESOP ’02)*, vol. 2305 of *Lecture Notes in Computer Science*, pp. 209–212, Springer, 2002.

[37] L. Chen, A. Miné, J. Wang, and P. Cousot, “Interval polyhedra: an abstract domain to inferinterval linear relationships,” in *Proceedings of the International Static Analysis Symposium (SAS ’09)*, vol. 5673 of *Lecture Notes in Computer Science*, pp. 309–325, Springer, 2009.

[38] L. Chen, A. Miné, J. Wang, and P. Cousot, “An abstract domain to discover interval linear equalities,” in *Proceedings of the International Conference on Verification, Model Checking, and Abstract Interpretation (VMCAI ’10)*, vol. 5944 of *Lecture Notes in Computer Science*, pp. 112–128, Springer, 2010.

[39] L. Chen, A. Miné, and P. Cousot, “A sound floating-point polyhedra abstract domain,” in *Proceedings of the Asian Symposium on Programming Languages and Systems (APLAS ’08)*, vol. 5356 of *Lecture Notes in Computer Science*, pp. 3–18, Springer, 2008.

[40] F. Ivančić, M. K. Ganai, S. Sankaranarayanan, and A. Gupta, “Numerical stability analysis of floating-point computations using software model checking,” in *Proceedings of the 8th ACM/IEEE International Conference on Formal Methods and

Models for Codesign (MEMOCODE '10), pp. 49–58, IEEE, July 2010.

[41] M. Brain, V. D'Silva, A. Griggio, L. Haller, and D. Kroening, "Deciding floating-point logic with abstract conflict driven clause learning," in *Formal Methods in System Design*, 2013.

[42] E. T. Barr, T. Vo, V. Le, and Z. Su, "Automatic detection of floating-point exceptions," in *Proceedings of the SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL '13)*, pp. 549–560, ACM, 2013.

[43] G. Barthe, J. M. Crespo, and C. Kunz, "Relational verification using product programs," in *Proceedings of the 17th International Symposium on Formal Methods (FM '11)*, pp. 200–214, Springer, 2011.

[44] M. Kovacs, H. Seidl, and B. Finkbeiner, "Relational abstract interpretation for the verification of 2-hypersafety properties," in *Proceedings of the ACM Conference on Computer and Communications Security (CCS '13)*, pp. 211–222, 2013.

Hindawi Publishing Corporation
Journal of Applied Mathematics
Volume 2014, Article ID 725275, 10 pages
http://dx.doi.org/10.1155/2014/725275

Research Article

Terminal Satisfiability in GSTE

Yongsheng Xu,[1] Guowu Yang,[1] Zhengwei Chang,[2] Desheng Zheng,[1] and Wensheng Guo[1]

[1] *School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu, Sichuan 611731, China*
[2] *State Grid Sichuan Electric Power Research Institute, Chengdu, Sichuan 610000, China*

Correspondence should be addressed to Guowu Yang; guowu@uestc.edu.cn

Received 9 February 2014; Accepted 10 March 2014; Published 15 May 2014

Academic Editor: Guiming Luo

Generalized symbolic trajectory evaluation (GSTE) is an extension of symbolic trajectory evaluation (STE) and a method of model checking. GSTE specifications are given as assertion graphs. There are four efficient methods to verify whether a circuit model obeys an assertion graph in GSTE, Model Checking Strong Satisfiability (SMC), Model Checking Normal Satisfiability (NMC), Model Checking Fair Satisfiability (FMC), and Model Checking Terminal Satisfiability (TMC). SMC, NMC, and FMC have been proved and applied in industry, but TMC has not. This paper gives a six-tuple definition and presents a new algorithm for TMC. Based on these, we prove that our algorithm is sound and complete. It solves the SMC's limitation (resulting in false negative) without extending from finite specification to infinite specification. At last, a case of using TMC to verify a realistic hardware circuit round-robin arbiter is achieved. Avoiding verifying the undesired paths which are not related to the specifications, TMC makes it possible to reduce the computational complexity, and the experimental results suggest that the time cost by SMC is 3.14× with TMC in the case.

1. Introduction

Logic errors found in finite state concurrent systems are extremely important problems for both circuit designers and programmers [1] (i.e., sequential circuit designs and communication protocols). Model checking is a technique for verifying finite state concurrent systems [1]. Symbolic trajectory evaluation is a lattice-based model checking technique based on a form of symbolic simulation [2–4]. STE has shown great promise in verifying medium to large scale industrial hardware designs with a high degree of automation at both the gate level and the transistor level [4–7]. *Generalized symbolic trajectory evaluation* [3, 8] is an extension of symbolic trajectory evaluation [9, 10]. STE is very limited in the types of properties that it can specify and verify. GSTE can handle ω*-regular* properties and maintain the efficiency and capacity of STE [9–13]. GSTE is originally developed at Intel and has been used successfully on Intel's next-generation microprocessors [14].

The main disadvantage of model checking is the state explosion, which motivates the need for algorithms such as abstract technology and the data structure of BDD and so forth to alleviate it [2, 15–18]. Although a lot of effort has been spent on improving this weakness, the efficient and effective method has yet to be developed. GSTE is a method of model checking and it also has the problem of state explosion. In GSTE, specifications are given as assertion graph and a model is induced by transition relation. Each edge in the model represents a state transition and a *trace* in the model is a state sequence, while a *path* in the assertion graph is an edge sequence. In SMC, NMC, and FMC, a model satisfies an assertion graph if all the traces (finite or infinite) in the model are accepted by all the paths of the same length in the assertion graph [3, 8, 19]. Results shown in [1, 9, 15, 17, 20–22] indicate that the method of enumeration usually leads to state explosion and increases the complexity of computing. As we know, traces are induced by model, and the number of traces is directly proportional to the size of model. Furthermore, paths are related to specifications. Thus, it will not need to verify each trace or path. Therefore, the key to alleviating the state explosion lies in deleting some undesired paths and traces. Aiming at this situation, this paper puts forward the filtering approach for TMC (Terminal Satisfiability Model Checking).

Previous studies have shown or suggested that a tool is needed to tell the difference between desired paths and undesired paths. One possible method is to choose E_T as the filter condition, where E_T is the set of edges in assertion graph. Our approach only adds some restricted conditions into assertion graphs without changing the structure of models and assertion graphs. Each edge in the assertion graph is labeled by an antecedent ant(e) and consequent cons(e). Ant(e) and cons(e) are sets of states in the model, so we can assign the states related to specifications to the ant(e) of edge which is in E_T. Then, we can delete undesired paths by E_T. A fuller discussion of TMC will appear in a later section.

In this paper, we focus on the need for alleviating state explosion and present a new algorithm for TMC based on GSTE. In Section 2, we introduce the basic definitions in GSTE. In Section 3, some concepts such as terminal assertion graph, terminal path, and terminal satisfiability are defined. After a statement of the basic concepts, related properties and a theorem are given. In Section 4, we present an algorithm for verifying terminal satisfiability with an example which cannot be verified by SMC but by TMC. In the end, we use TMC to successfully verify a hardware circuit round-robin arbiter, and the time cost by SMC is 3.14× with TMC in the case. In Section 5, we prove that our algorithm is sound and complete. Section 6 is the conclusion of the paper.

2. Preliminaries

We introduce some basic definitions in GSTE [3, 8]. We assume a universal set of finite states, denoted by S.

2.1. Model

Definition 1 (transition relation; see [3, 8]). A relation $T \subseteq S \times S$ is a transition relation if for all $s \in S$, $\exists s' \in S$, $(s, s') \in T$.

Definition 2 (model; see [3, 8]). The model M induced by the transition relation T is the pair (pre, post), where

(1) the preimage transformer pre: $2^S \rightarrow 2^S$ is defined as

$$\text{pre}(Q) = \left\{s \mid \exists s' \in Q, \left(s, s'\right) \in T\right\} \quad \forall Q \in 2^S, \tag{1}$$

(2) and the postimage transformer post: $2^S \rightarrow 2^S$ is defined as

$$\text{post}(Q) = \left\{s' \mid \exists s \in Q, \left(s, s'\right) \in T\right\} \quad \forall Q \in 2^S. \tag{2}$$

Definition 3 (trace; see [3, 8]). A trace σ in model M = (pre, post) is a state sequence such that $\sigma[i+1] \in \text{post}(\sigma[i])$ for all $1 \le i \le |\sigma|$.

2.2. Assertion Graph

Definition 4 (assertion graph; see [3, 8]). An assertion graph is a quintuple $G = (V, v_0, E, \text{ant}, \text{cons})$, where V is a finite set of vertices, v_0 is the initial vertex, $E \subseteq V \times V$ is a set of edges satisfying for all $u \in V$, $\exists v \in V$, $(u, v) \in E$, ant is a mapping: $E \rightarrow 2^S$, and cons is a mapping: $E \rightarrow 2^S$.

Definition 5 (path; see [3, 8]). A path ρ in assertion graph G is an edge sequence such that $\rho[i]$ ends at a vertex from which $\rho[i+1]$ starts for all $1 \le i \le |\rho|$.

Definition 6 (a trace is accepted by a path; see [3, 8]). An execution trace σ in a model is accepted by a path ρ in an assertion graph if (suppose the length of trace and path is n)

$$\forall i_{1\le i\le n} \sigma[i] \in \text{ant}(\rho[i]) \Longrightarrow \forall i_{1\le i\le n} \sigma[i] \in \text{cons}(\rho[i]), \tag{3}$$

where $\sigma[i]$ is the ith state of the trace σ and $\rho[i]$ is the ith edge of the path ρ.

2.3. Four Kinds of Verifying Algorithms. Based on Definition 6, GSTE has four kinds of acceptance [19].

Definition 7 (SMC (model checking strong satisfiability)). A model M strongly satisfies an assertion graph G if for all finite initial paths ρ in G and all finite traces σ in M of the same length σ is accepted by ρ, denoted by $M \models G$.

Definition 8 (NMC (model checking normal satisfiability)). A model M normally satisfies an assertion graph G if for every infinite initial path ρ_ω in G and every infinite trace σ_ω in M of the same length σ_ω is accepted by ρ_ω, denoted by $M \models G$.

Definition 9 (FMC (model checking fair satisfiability)). A model M fairly satisfies an assertion graph G under F (fairness constraint) if for every infinite fair path ρ_ω in G and every infinite trace σ_ω in M, σ_ω is accepted by ρ_ω, denoted by $M \models_F G$.

Definition 10 (TMC (model checking terminal satisfiability)). A model M terminally satisfies an assertion graph G under E_T (terminal edge set) if for all finite terminal paths ρ_T in G and all finite traces σ in M of the same length σ is accepted by ρ_T, denoted by $M \models_T G$.

We use the following notations in the rest of the paper, $\rho = [e_1, e_2, \ldots]$ be an arbitrary sequence of elements, $|\rho|$ be the length of the sequence, and $\rho[i]$ be the ith element e_i in the sequence. Further, $(\rho : e)$ denotes the sequence by appending element e to the end of ρ when ρ is finite, and $(e : \rho)$ is the sequence by appending element e to the head of ρ.

3. Terminal Satisfiability

In this section, we define some concepts such as terminal assertion graph and terminal path and give some related properties on them.

In GSTE, the set of states in model M is finite ($|S|$ is finite), but each state has its successor states so that traces in the model can be extended to infinite length. Each edge in the assertion graph has its successor edges so that assertion graphs can specify infinite properties. It should be noted that in our daily life, most of the properties which have been verified are finite rather than infinite. In particular, some properties are terminated at some states and these states are not involved. In GSTE existing theories, the SMC algorithm is the only one used to handle finite properties [3, 8]; it cannot stop at some edges whose labels ant(e) are not related to the specifications and there are many redundant calculations. The program proves that we can predict the behavior of the model; these observations lead us to hypothesize that finite specifications will terminate at some states which we can predict. Corresponding to terminal specifications, the paths will also terminate at some edges, which are named terminal edge. Our algorithm will make the labels ant(e) on the edges, which cannot reach terminal edges, equal to $\emptyset$. The method outlined here can be used to reduce the complexity of computation.

For convenience, we expanded the assertion graph into six-tuple.

Definition 11 (terminal assertion graph). A terminal assertion graph is a six-tuple $G = (V, v_0, E, \text{ant}, \text{cons}, E_T)$, where V, v_0, E, ant, and cons are the same as in preliminaries; $E_T \subseteq E$ is a set of terminal edges.

Definition 12 (terminal path). A terminal path is

$$\rho_T = e_0 e_1 e_2 \cdots e_{|\rho_T|} \quad (e_i \in E,\ i = 0, 1, 2, \ldots, |\rho_T|), \tag{4}$$

where e_0 is an initial edge, it starts from v_0, and $e_{|\rho_T|}$ is a terminal edge, $e_{|\rho_T|} \in E_T$. The last edge of path ρ_T must be terminal edge.

Definition 13 (a trace is accepted by a terminal path; see [3, 8, 19]). Let $G = (V, v_0, E, \text{ant}, \text{cons}, E_T)$ be a terminal assertion graph and $M = (\text{pre}, \text{post})$ be a model. Given an edge labeling $\gamma : E \rightarrow 2^S$ where γ is either ant or cons, a trace σ in M satisfies a terminal path ρ_T of the same length under γ if for every $1 \le i \le |\sigma|$, $\sigma[i] \in \gamma(\rho_T[i])$, denoted by

$$(M, \sigma) \vDash_\gamma (G, \rho_T) \quad (\gamma = \text{ant or cons}). \tag{5}$$

The trace σ is accepted by the terminal path ρ_T if

$$(M, \sigma) \vDash_{\text{ant}} (G, \rho_T) \Longrightarrow (M, \sigma) \vDash_{\text{cons}} (G, \rho_T), \tag{6}$$

denoted by $(M, \sigma) \vDash_T (G, \rho_T)$.

Definition 14 (model satisfies terminal assertion graph). A model M satisfies a terminal assertion graph G if for all terminal paths ρ_T in G and all traces σ in M of the same length one has $(M, \sigma) \vDash_T (G, \rho_T)$, denoted by

$$M \vDash_T G. \tag{7}$$

Definition 15 (model strongly satisfies assertion graph; see [3, 8]). A model M strongly satisfies an assertion graph G ($E_T = \emptyset$) if for all finite paths ρ in G and all traces σ in M of the same length one has $(M, \sigma) \vDash (G, \rho)$, denoted by

$$M \vDash G. \tag{8}$$

The Difference between Finite Path and Terminal Path. Note that there is a key difference between finite path and terminal path; unlike terminal path, which must be end up with a terminal edge $e \in E_T$, finite path can end up with any edge $e \in E$. Given our analysis, this suggests a terminal path will be a finite path; however, a finite path may not be a terminal path.

Let $A = \{\rho \mid \rho$ is a finite initial path and the length of ρ is $|\rho|$, where $|\rho| = 1, 2, \ldots\}$ and $B = \{\rho_T \mid \rho_T$ is a terminal path and the length of ρ_T is $|\rho_T|$, where $|\rho_T| = 1, 2, \ldots\}$. The last edge of ρ_T must be a terminal edge, $\rho_T[|\rho_T|] \in E_T$; however, the last edge of ρ does not have this restriction, it just needs to meet $\rho[|\rho|] \in E$, $E_T \subseteq E$, so one has $B \subseteq A$.

Theorem 16. *For any assertion graph $G = (V, v_0, E, ant, cons, E_T)$ and any model M, one has*

$$M \vDash G \Longrightarrow M \vDash_T G. \tag{9}$$

Proof. According to Definition 15

$$\begin{aligned} M \vDash G &\Longleftrightarrow \forall \rho \in A, \quad (M, \sigma) \vDash (G, \rho) \\ &\Longrightarrow \forall \rho_T \in B, \quad (M, \sigma) \vDash_T (G, \rho_T) \quad (\text{using } B \subseteq A) \\ &\Longleftrightarrow M \vDash_T G \quad (\text{using Definition } 14). \end{aligned} \tag{10}$$

□

Theorem 16 describes the relationship between SMC and TMC. If SMC returns true, then TMC returns true, but not vice versa. When $E_T = E$, terminal satisfiability behaves as strong satisfiability. When $E_T \subseteq E$, $E_T \neq E$, we can remove some undesired paths in the assertion graph.

4. Model Checking with Terminal Satisfiability

In this section, we describe a model checking method for verifying terminal satisfiability. The concept of terminal satisfiability was proposed in [19] but was not given any algorithm. In this paper, we present an algorithm and prove it.

4.1. The TMC Algorithm. In [3, 8], the algorithms for SMC, NMC, and FMC were given. Our TMC algorithm is similar. First, we should define the backward simulation sequence.

Definition 17. Given an assertion graph $G = (V, v_0, E, \text{ant}, \text{cons}, E_T)$ and a model $M = (\text{pre}, \text{post})$, the backward simulation sequence is as below

$$[\varphi_1, \varphi_2, \varphi_3, \ldots], \tag{11}$$

where $\varphi_n : E \rightarrow 2^S$ $(n \ge 1)$ is the n-step backward simulation relation. It is defined as

$$\varphi_1(e) = \begin{cases} \displaystyle\bigcup_{e^+ \in \text{out}(e) \cap E_T} (\text{pre}(\text{ant}(e^+)) \cap \text{ant}(e)) & \forall e \in E - E_T, \\ \text{ant}(e) & \forall e \in E_T, \end{cases}$$

$$\varphi_n(e) = \begin{cases} \varphi_n(e) \cup \left(\displaystyle\bigcup_{e^+ \in \text{out}(e)} (\text{pre}(\varphi_n(e^+)) \cap \text{ant}(e)) \right) & \forall e \in E - E_T, \\ \text{ant}(e) & \forall e \in E_T. \end{cases} \tag{12}$$

Based on this result, when we verify finite properties, we simply verify those traces that can reach some states in ant(e), where $e \in E_T$. These states, which are related to the property, are assigned to the label ant of the terminal edge. Then we emulate the model's behavior backwards starting from the terminal edge. The n-step backward simulation relation φ_n is the set of the states that can reach the terminal edge within n-step.

Obviously, the backward simulation relation sequence is nondecreasing and has a least fix-point upper limit bounded by ant. We denote the least fix-point by φ_T^*. Now we propose a terminal satisfiability model checking algorithm that first performs a series of graph transformations and then checks strong satisfiability on the resulting graph.

4.2. The Application of TMC. In order to describe TMC clearly, let us look at the following example [23].

Example 18. Consider the model M in Figure 1, where $S = \{s_0, s_1, s_2, s_3, s_4, s_5, s_6\}$.

Suppose we want to specify the following property: if the system is in the state s_5, then it must have been in s_3 the last time when it was neither in s_5 nor s_6. The model satisfies this property. Now, let us catch this property using the assertion graph in Figure 2.

The fact that SMC cannot verify this property is due to undesired paths, such as the finite initial path $\rho_1 = [(v_0, v_1)]$ and the finite trace $\sigma_1 = [s_4]$, $\sigma_1 \vDash_{\text{ant}} \rho_1$, but the first state s_4 of σ_1 is not in the cons((v_0, v_1)) $= \{s_3\}$. Thus, the model does not strongly satisfy the assertion graph. If we use SMC, the SMC algorithm will return false. In order to deal with this situation in the former, [3, 8] introduce NMC and FMC. As we know, NMC and FMC have been used to deal with *ω-regular* properties. If we want to solve the SMC's limitation, we should expand finite specifications to infinite specifications. This is the key that leads to the increasing complexity of the model checking. However, the algorithm of using TMC to verify this property is more suitable. TMC can be accomplished without expanding finite specifications to infinite specifications. The following example is constructed only for the purpose of illustrating the computational procedure discussed.

Step 1. Construct the terminal assertion graph and figure out the terminal edges (Figure 3).

Step 2. First, the fix-point φ_T^* of every edge is computed according to Definition 17; then the label ant of every edge is replaced by the corresponding φ_T^*; lastly, the resulting assertion graph G' is gotten.

Step 3. This step behaves as the algorithm SMC : SMC(M, G'). The edge which is marked by the blue line is the terminal edge. Supposing $E_T = \{(v_1, v_2)\}$, there is only one terminal edge in the case (Figure 3). Thus, the terminal path can be grouped into two cases (as follows):

$$\rho_{T_1} = [(v_0, v_1)(v_1, v_2)] \left(\left|\rho_{T_1}\right| = 2\right),$$
$$\rho_{T_2} = [(v_0, v_1)(v_1, v_1)\cdots(v_1, v_1)(v_1, v_2)] \left(\left|\rho_{T_2}\right| \geq 3\right). \tag{13}$$

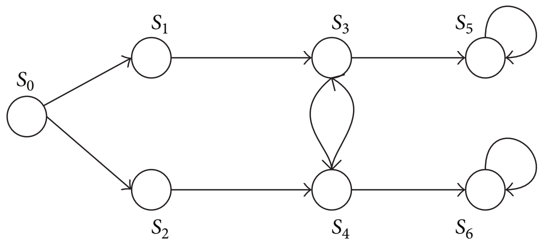


FIGURE 1: A simple model.

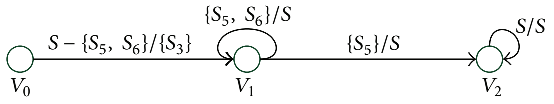


FIGURE 2: An assertion graph.

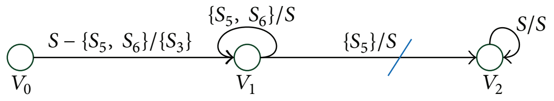


FIGURE 3: A terminal assertion graph.

Because the terminal path must end up with the terminal edge, there is only one terminal edge in Figure 3 and the vertex v_2 to v_1 is unreachable. Thus, if a path reaches the edge (v_2, v_2) that is not a terminal path, then all terminal paths are finite. As we know, the length of a terminal path must be greater than or equal to 2. Thus, we can eliminate the path ρ_1 and the trace σ_1, which will lead to SMC returning false negative, since $|\rho_1| = |\sigma_1| = 1$. The trace in model M (Figure 1), if not starting from s_3, ultimately would violate the label ant((v_1, v_2)) on the edge (v_1, v_2); therefore the model M (Figure 1) terminally satisfies the terminal assertion graph G (Figure 3). The calculation is as follows (after one iteration, every edge can be reached to the fix-point).

The first iteration (initialization):

$$\begin{aligned}
\varphi_1(v_1, v_2) &= \text{ant}((v_1, v_2)) = \{s_5\}, \\
\varphi_1(v_1, v_1) &= \text{pre}(\text{ant}((v_1, v_2))) \cap \text{ant}((v_1, v_1)) \\
&= \text{pre}(\{s_5\}) \cap \{s_5, s_6\} = \{s_3, s_5\} \cap \{s_5, s_6\} = \{s_5\}, \\
\varphi_1(v_0, v_1) &= \text{pre}(\text{ant}((v_1, v_2))) \cap \text{ant}((v_0, v_1)) \\
&= \{s_3, s_5\} \cap \{s_0, s_1, s_2, s_3, s_4\} \\
&= \{s_3\}, \\
\varphi_1(v_2, v_2) &= \emptyset.
\end{aligned} \tag{14}$$

The second iteration:

$$\begin{aligned}
\varphi_2(v_1, v_2) &= \{s_5\}, \\
\varphi_2(v_1, v_1) &= \varphi_1(v_1, v_1) \cup (\text{pre}(\varphi_1(v_1, v_1)) \cap \text{ant}((v_1, v_1))) \\
&\cup (\text{pre}(\varphi_1(v_1, v_2)) \cap \text{ant}((v_1, v_1))) \\
&= \{s_5\} \cup (\text{pre}\{s_5\} \cap \{s_5, s_6\}) \\
&= \{s_5\},
\end{aligned}$$

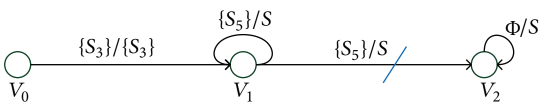


FIGURE 4: The resulting assertion graph G'.

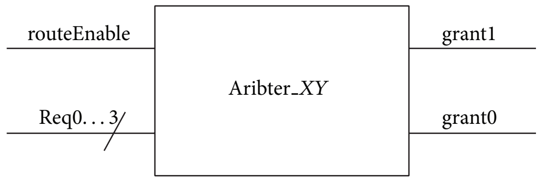


FIGURE 5: An arbiter with 4 inputs.

$$\begin{aligned}\varphi_2(v_0,v_1) &= \varphi_1(v_0,v_1)\cup(\mathrm{pre}(\varphi_1(v_1,v_1))\cap\mathrm{ant}((v_0,v_1)))\\ &\cup\ (\mathrm{pre}(\varphi_1(v_1,v_2))\cap\mathrm{ant}((v_0,v_1)))\\ &= \{s_3\}\cup(\mathrm{pre}\{s_5\}\cap\{s_0,s_1,s_2,s_3,s_4\})\\ &\cup\ (\mathrm{pre}\{s_5\}\cap\{s_0,s_1,s_2,s_3,s_4\})\\ &= \{s_3\},\\ \varphi_2(v_2,v_2) &= \emptyset.\end{aligned} \tag{15}$$

After replacing the label ant of each edge with the corresponding fix-point φ_T^*, the resulting assertion graph G' is shown as Figure 4.

As can be seen, ant of each edge has been reached by the minimum, and TMC returns true. Thus, the model satisfies the terminal assertion graph.

4.3. A Hardware Verification Case of TMC

4.3. A Hardware Verification Case of TMC. In this section, we present a case study of TMC on hardware verification. The circuit we choose for this study is a round-robin arbiter, which is the core component in many real network systems [24]. An arbiter has N inputs and the output is a vector encoding of arbitration results. Request with highest priority in round-robin order is granted in each cycle, and the priority of the request which is granted in last arbitration will become the lowest in the next round; this protocol guarantees a dynamic priority assignment to requestors without starvation. We consider the arbiter with 4 inputs (shown in Figure 5). First, the priority of inputs is placed in descending order from req[0] to req[3] [25, 26]. Thus, req[0] has the highest priority, req[1] has the next priority, and so on. Each input of the arbiter in Figure 5 is connected to a switch cell; only one of the four inputs will succeed each clock cycle. Others are rejected and must retry later. routeEnable is enable signal.

We model this arbiter as a finite state machine (shown in Figure 6) according to its truth table in paper [24]. The input req[3 : 0] is a vector of ternary-valued variables; each variable can take 0, 1, or X. X indicates that variable can take 0 or 1. Figure 6 is the model of this arbiter. Consider

$$\begin{aligned}&\mathrm{trans\_0\_0} = \mathrm{req1} = 0\&\mathrm{req2} = 0\&\mathrm{req3} = 0\&\mathrm{routeEnable} = 1,\\ &\mathrm{trans\_0\_1} = \mathrm{req1} = 1\&\mathrm{ruoteEnable} = 1,\\ &\mathrm{trans\_0\_2} = \mathrm{req1} = 0\&\mathrm{req2} = 1\&\mathrm{routeEnable} = 1,\\ &\mathrm{trans\_0\_3} = \mathrm{req1} = 0\&\mathrm{req2} = 0\&\mathrm{req3} = 1\&\mathrm{routeEnable} = 1,\\ &\mathrm{trans\_1\_1} = \mathrm{req2} = 0\&\mathrm{req3} = 0\&\mathrm{req0} = 0\&\mathrm{routeEnable} = 1,\\ &\mathrm{trans\_1\_2} = \mathrm{req2} = 1\&\mathrm{routeEnable} = 1,\\ &\mathrm{trans\_1\_3} = \mathrm{req2} = 0\&\mathrm{req3} = 1\&\mathrm{routeEnable} = 1,\\ &\mathrm{trans\_1\_0} = \mathrm{req2} = 0\&\mathrm{req3} = 0\&\mathrm{req0} = 1\&\mathrm{routeEnable} = 1,\\ &\mathrm{trans\_2\_2} = \mathrm{req3} = 0\&\mathrm{req0} = 0\&\mathrm{req1} = 0\&\mathrm{routeEnable} = 1,\\ &\mathrm{trans\_2\_3} = \mathrm{req3} = 1\&\mathrm{routeEnable} = 1,\\ &\mathrm{trans\_2\_0} = \mathrm{req3} = 0\&\mathrm{req0} = 1\&\mathrm{routeEnable} = 1,\\ &\mathrm{trans\_2\_1} = \mathrm{req3} = 0\&\mathrm{req0} = 0\&\mathrm{req1} = 1\&\mathrm{routeEnable} = 1,\\ &\mathrm{trans\_3\_3} = \mathrm{req0} = 0\&\mathrm{req1} = 0\&\mathrm{req2} = 0\&\mathrm{routeEnable} = 1,\\ &\mathrm{trans\_3\_0} = \mathrm{req0} = 1\&\mathrm{routeEnable} = 1,\\ &\mathrm{trans\_3\_1} = \mathrm{req0} = 0\&\mathrm{req1} = 1\&\mathrm{routeEnable} = 1,\\ &\mathrm{trans\_3\_2} = \mathrm{req0} = 0\&\mathrm{req1} = 0\&\mathrm{req2} = 1\&\mathrm{routeEnable} = 1.\end{aligned} \tag{16}$$

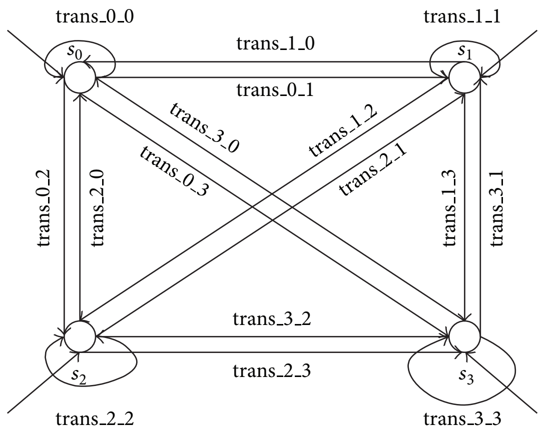


FIGURE 6: The behavior of arbiter.

Safety property: once a request reqi is set high from a state and kept high, then the request will be granted after several cycles. Suppose a state where the value of grant is [1, 0], which means that the last request granted is req2; if the request req2 is set high again and kept high, then the request will be granted after at most 4 cycles. We use the terminal assertion graph to specify this property (shown as in Figure 7) [24]. Consider

$$\begin{aligned}&\mathrm{ant\_}V_I\mathrm{\_}V\mathrm{set} = \mathrm{req0} = 0\&\mathrm{req1} = 0\&\mathrm{req2} = 1\&\mathrm{req3} = 0,\\ &\mathrm{ant\_}V\mathrm{set\_}V_1 = \mathrm{req2} = 1\&\mathrm{req3} = 0\&\mathrm{req0} = 0\&\mathrm{req1} = 1,\\ &\mathrm{ant\_}V\mathrm{set\_}V_3 = \mathrm{req2} = 1\&\mathrm{req3} = 1,\\ &\mathrm{ant\_}V\mathrm{set\_}V_0 = \mathrm{req2} = 1\&\mathrm{req3} = 0\&\mathrm{req0} = 1,\\ &\mathrm{ant\_}V_1\mathrm{\_}V_2 = \mathrm{req2} = 1,\end{aligned}$$

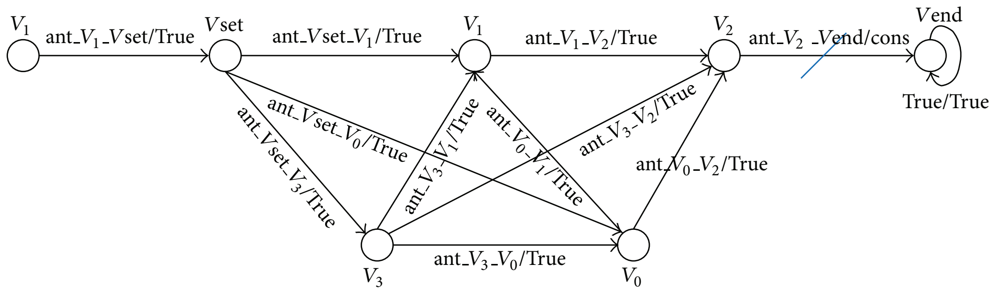


FIGURE 7: The terminal assertion graph.

$$\begin{aligned}
\text{ant\_}V_0\text{\_}V_1 &= \text{req2} = 1\&\text{req1} = 1,\\
\text{ant\_}V_0\text{\_}V_2 &= \text{req2} = 1\&\text{req1} = 0,\\
\text{ant\_}V_3\text{\_}V_0 &= \text{req2} = 1\&\text{req0} = 1,\\
\text{ant\_}V_3\text{\_}V_1 &= \text{req2} = 1\&\text{req0} = 0\&\text{req1} = 1,\\
\text{ant\_}V_3\text{\_}V_2 &= \text{req2} = 1\&\text{req0} = 0\&\text{req1} = 0,\\
\text{ant\_}V_2\text{\_}V\text{end} &= \text{grant1} = 0\&\text{grant0} = 1,\\
\text{cons} &= \text{grant1} = 0\&\text{grant0} = 1.
\end{aligned} \tag{17}$$

The edge $(V_2, V\text{end})$ is the terminal edge. We only verify those paths that end up with the edge $(V_2, V\text{end})$. If we use SMC to verify this property, we must verify all finite paths. If using TMC, we only need to verify terminal paths. In the terminal assertion graph (Figure 7), all finite paths are as follows:

$$\begin{aligned}
\rho_1 &= [V_I, V\text{set}],\\
\rho_2 &= [V_I, V\text{set}][V\text{set}, V_1],\\
\rho_3 &= [V_I, V\text{set}][V\text{set}, V_3],\\
\rho_4 &= [V_I, V\text{set}][V\text{set}, V_0],\\
\rho_5 &= [V_I, V\text{set}][V\text{set}, V_1][V_1, V_2],\\
\rho_6 &= [V_I, V\text{set}][V\text{set}, V_3][V_3, V_1],\\
\rho_7 &= [V_I, V\text{set}][V\text{set}, V_0][V_0, V_1],\\
\rho_8 &= [V_I, V\text{set}][V\text{set}, V_3][V_3, V_0],\\
\rho_9 &= [V_I, V\text{set}][V\text{set}, V_3][V_3, V_2],\\
\rho_{10} &= [V_I, V\text{set}][V\text{set}, V_3][V_3, V_1][V_1, V_2],\\
\rho_{11} &= [V_I, V\text{set}][V\text{set}, V_3][V_3, V_0][V_0, V_1],\\
\rho_{12} &= [V_I, V\text{set}][V\text{set}, V_3][V_3, V_0][V_0, V_1][V_1, V_2],\\
\rho_{13} &= [V_I, V\text{set}][V\text{set}, V_3][V_3, V_0][V_0, V_2],\\
\rho_{14} &= [V_I, V\text{set}][V\text{set}, V_0][V_0, V_1][V_1, V_2],\\
\rho_{15} &= [V_I, V\text{set}][V\text{set}, V_0][V_0, V_2],\\
\rho_{16} &= [V_I, V\text{set}][V\text{set}, V_1][V_1, V_2][V_2, V\text{end}],\\
\rho_{17} &= [V_I, V\text{set}][V\text{set}, V_0][V_0, V_1][V_1, V_2][V_2, V\text{end}],\\
\rho_{18} &= [V_I, V\text{set}][V\text{set}, V_0][V_0, V_2][V_2, V\text{end}],\\
\rho_{19} &= [V_I, V\text{set}][V\text{set}, V_3][V_3, V_1][V_1, V_2][V_2, V\text{end}],\\
\rho_{20} &= [V_I, V\text{set}][V\text{set}, V_3][V_3, V_2][V_2, V\text{end}],\\
\rho_{21} &= [V_I, V\text{set}][V\text{set}, V_3][V_3, V_0][V_0, V_2][V_2, V\text{end}],\\
\rho_{22} &= [V_I, V\text{set}][V\text{set}, V_3][V_3, V_0][V_0, V_1][V_1, V_2]\\
&\quad \times [V_2, V\text{end}].
\end{aligned} \tag{18}$$

If we use SMC, we must verify 22 paths: $\rho_1, \ldots, \rho_{22}$. But we use TMC only to verify $\rho_{16}, \ldots, \rho_{22}$. Therefore, the time cost by SMC is 3.14× with TMC in this case.

5. Correctness Proof

In this section, we formally prove that the TMC algorithm is both sound and complete.

5.1. The Logic of Algorithm. $M \models_T G \Leftrightarrow$ for all ρ_T in G, for all σ in M of the same length, and $(M, \sigma) \models_T (G, \rho_T) \Leftrightarrow$ for all ρ_T in G and for all σ in M of the same length,

$$(M, \sigma) \models_{\text{ant}} (G, \rho_T) \Longrightarrow (M, \sigma) \models_{\text{cons}} (G, \rho_T). \tag{19}$$

By analyzing algorithm TMC, in the end, we use SMC(M, G'). Consider

$$\text{SMC}(M, G') \text{ returns true} \Longrightarrow M \models G'. \tag{20}$$

It indicates that we can only get

$$\begin{gathered}
\forall \rho \text{ in } G, \quad \forall \sigma \text{ in } M \text{ of the same length,}\\
(M, \sigma) \models_{\varphi_T^*} (G, \rho) \Longrightarrow (M, \sigma) \models_{\text{cons}} (G, \rho).
\end{gathered} \tag{21}$$

But we are aiming at (according to (19))

$$\begin{gathered}
\forall \rho_T \text{ in } G, \quad \forall \sigma \text{ in } M \text{ of the same length,}\\
(M, \sigma) \models_{\text{ant}} (G, \rho_T) \Longrightarrow (M, \sigma) \models_{\text{cons}} (G, \rho_T).
\end{gathered} \tag{22}$$

Then, if we want to prove TMC is correct, we must prove that (according to (21) and (22))

$$\begin{gathered}\forall\rho_T \text{ in } G, \quad \forall\sigma \text{ in } M \text{ of the same length,}\\ (M,\sigma)\vDash_{\text{ant}}(G,\rho_T) \Longrightarrow (M,\sigma)\vDash_{\varphi_T^*}(G,\rho_T).\end{gathered} \tag{23}$$

5.2. The TMC Algorithm Is Sound. According to Definition 17, we get the following lemma.

Lemma 19. *For a terminal assertion graph $G = (V, v_0, E, ant, cons, E_T)$ and a model $M = (pre, post)$, let $[\varphi_1, \varphi_2, \varphi_3, \ldots]$ be the backward simulation sequence towards $E_T \subseteq E$. Then for any $n \geq 2$, for all finite terminal paths ρ_T, and all finite traces σ of length $n+1$, such that*

(1) $\rho_T[n+1] \in E_T$

(2) *when $\sigma^2 \vDash_{ant} \rho_T^2$ and $|\sigma^2| = |\rho_T^2| = n$, we have $\sigma[i] \in \varphi_{n+1-i}(\rho_T[i])$ $(i = 2, 3, \ldots, n)$ (note: ρ_T^2 represents all paths of length n which are obtained by traversing backwards starting from the terminal edge. For example, if $\rho_T = e_1e_2\cdots e_ne_{n+1}$ and $e_{n+1} \in E_T$, then $\rho_T^2 = e_2e_3\cdots e_ne_{n+1}$, σ^2 represents all traces of length n obtained by traversing backwards starting from the last state of σ)*

(3) *when $\sigma \vDash_{ant} \rho_T$, we have $\sigma[1] \in \varphi_n(\rho_T[1])$, $\sigma[n+1] \in \varphi_1(\rho_T[n+1])$.*

Proof (mathematical induction). Suppose $|\rho_T| = |\sigma| = k+1$.

First Step ($k = 2$). (1) $\rho_T[3] \in E_T$ since ρ_T is a terminal path.

(2) Let us prove $\sigma[2] \in \varphi_1(\rho_T[2])$:

$$\sigma^2 \vDash_{\text{ant}} \rho_T^2 \Longrightarrow \sigma[2] \in \text{ant}(\rho_T[2]), \quad \sigma[3] \in \text{ant}(\rho_T[3]), \tag{24}$$

$$\sigma[2] \in \text{pre}(\sigma[3]) \overset{(24)}{\Longrightarrow} \sigma[2] \in \text{pre}(\text{ant}(\rho_T[3])). \tag{25}$$

According to (24)-(25),

$$\sigma[2] \in \text{ant}(\rho_T[2]) \cap \text{pre}(\text{ant}(\rho_T[3])) \subseteq \varphi_1(\rho_T[2]). \tag{26}$$

(3) Let us prove $\sigma[1] \in \varphi_2(\rho_T[1])$, $\sigma[3] \in \varphi_1(\rho_T[3])$:

$$\sigma[1] \in \text{pre}(\sigma[2]) \overset{(26)}{\Longrightarrow} \sigma[1] \in \text{pre}(\varphi_1(\rho_T[2])) \tag{27}$$

$$\sigma \vDash_{\text{ant}} \rho_T \Longrightarrow \sigma[1] \in \text{ant}(\rho_T[1]). \tag{28}$$

According to (27)-(28),

$$\begin{aligned}&\sigma[1] \in \text{ant}(\rho_T[1]) \cap \text{pre}(\varphi_1(\rho_T[2]))\\ &\quad\subseteq \varphi_1(\rho_T[1])\\ &\quad\cup\left(\bigcup_{e^+\in\text{out}(\rho_T[1])}(\text{pre}(\varphi_1(e^+)) \cap \text{ant}(\rho_T[1]))\right) \qquad (29)\\ &\quad= \varphi_2(\rho_T[1]),\\ &\sigma \vDash_{\text{ant}} \rho_T \Longrightarrow \sigma[3] \in \text{ant}(\rho_T[3]) = \varphi_1(\rho_T[3]).\end{aligned}$$

Then, the conclusions (1), (2), and (3) are true for $k = 2$.

Second Step. We already saw that the conclusions are true for $k = 2$. Assume this, for an arbitrary n, the above lemma is true for $k = n$, then we get the following.

(σ represents all traces of length $n+1$ and ρ_T represents all terminal paths of the same length) (1) $\rho_T[n+1] \in E_T$. (2) When $\sigma^2 \vDash_{\text{ant}} \rho_T^2$, we have $\sigma[i] \in \varphi_{n+1-i}(\rho_T[i])$ $(i = 2, 3, \ldots, n)$ (where ρ_T^2 represents all paths of length k obtained by traversing backwards starting from the terminal edge; the terminal edge is the last edge of the terminal path; σ^2 is similar), (3) when $\sigma \vDash_{\text{ant}} \rho_T$, we have $\sigma[1] \in \varphi_n(\rho_T[1])$, $\sigma[n+1] \in \varphi_1(\rho_T[n+1])$.

Third Step (let us derive the lemma is true for $k = n+1$ from above assumption). We must consider all terminal paths ρ_T of length $n+2$ and all traces of the same length.

(1) $\rho_T[n+2] \in E_T$ since ρ_T is a terminal path.

(2) We must prove $\sigma^2 \vDash_{\text{ant}} \rho_T^2 \Rightarrow \sigma[i] \in \varphi_{n+2-i}(\rho_T[i])$ $(i = 2, 3, \ldots, n, n+1)$ at this step, $|\rho_T^2| = |\sigma^2| = n+1$

$$\sigma^2 \vDash_{\text{ant}} \rho_T^2 \Longrightarrow \sigma^3 \vDash_{\text{ant}} \rho_T^3, \tag{30}$$

where ρ_T^3 represents all paths of length n obtained by traversing starting from the terminal edge and σ^3 is obtained by deleting the first two states of σ, $|\sigma| = n+2 \Rightarrow |\sigma^3| = n$, and $|\rho_T| = n+2 \Rightarrow |\rho_T^3| = n$. By the induction hypothesis we have the following conclusion:

$$\begin{gathered}\sigma^3 \vDash_{\text{ant}} \rho_T^3 \Longrightarrow \sigma[i] \in \varphi_{n+1-i}(\rho_T[i]) \subseteq \varphi_{n+2-i}(\rho_T[i]),\\ (i = 3, 4, \ldots, n, n+1).\end{gathered} \tag{31}$$

According to (31),

$$\sigma[3] \in \varphi_{n-1}(\rho_T[3]), \tag{32}$$

$$\sigma[2] \in \text{pre}(\sigma[3]) \overset{(32)}{\Longrightarrow} \sigma[2] \in \text{pre}(\varphi_{n-1}(\rho_T[3])), \tag{33}$$

$$\begin{aligned}\sigma[2] \in \text{ant}(\rho_T[2]) \overset{(33)}{\Longrightarrow}&\ \sigma[2] \in \text{ant}(\rho_T[2])\\ &\cap \text{pre}(\varphi_{n-1}(\rho_T[3]))\\ &\subseteq \varphi_{n-1}(\rho_T[2])\\ &\cup\left(\bigcup_{e^+\in\text{out}(\rho_T[2])}(\text{pre}(\varphi_{n-1}(e^+)) \cap \text{ant}(\rho_T[2]))\right)\\ &= \varphi_n(\rho_T[2]),\end{aligned} \tag{34}$$

namely, $\sigma^2 \vDash_{\text{ant}} \rho_T^2 \Rightarrow \sigma[i] \in \varphi_{n+2-i}(\rho_T[i])$ $(i = 2, 3, \ldots, n, n+1)$.

(3) Consider

$$\sigma \vDash_{\text{ant}} \rho_T \Longrightarrow \sigma[1] \in \text{ant}(\rho_T[1]), \tag{35}$$

$$\sigma \vDash_{\text{ant}} \rho_T \Longrightarrow \sigma^2 \vDash_{\text{ant}} \rho_T^2 \overset{(34)}{\Longrightarrow} \sigma[2] \in \varphi_n(\rho_T[2]), \tag{36}$$

$$\sigma[1] \in \text{pre}(\sigma[2]) \overset{(36)}{\Longrightarrow} \sigma[1] \in \text{pre}(\varphi_n(\rho_T[2])). \tag{37}$$

According to (35)–(37),

$$\begin{aligned}\sigma[1] &\in \operatorname{ant}(\rho_T[1]) \cap \operatorname{pre}(\varphi_n(\rho_T[2])) \\ &\subseteq \varphi_n(\rho_T[1]) \\ &\cup \left(\bigcup_{e^+ \in \operatorname{out}(\rho_T[1])} (\operatorname{pre}(\varphi_n(e^+)) \cap \operatorname{ant}(\rho_T[1]))\right) \\ &= \varphi_{n+1}(\rho_T[1]), \\ \sigma \vDash_{\operatorname{ant}} \rho_T &\Longrightarrow \sigma[n+2] \in \operatorname{ant}(\rho_T[n+2]) = \varphi_1(\rho_T[n+2])\end{aligned} \tag{38}$$

which exactly means that the lemma holds for $k = n+1$. Therefore, the lemma is true for all n starting with 2. □

The conclusions of Lemma 19 have been used to prove Theorem 20. At the same time, Lemma 19 gives the conclusion that the backward simulation sequence φ_n is a convergent sequence. If the length of the path is n, then the simulation sequence of the ith edges will reach the fixed point through at least $n+1-i$ iterations.

Suppose φ_T^* is the fix-point of the backward simulation sequence based on the terminal set E_T; after replacing the label ant of each edge with the corresponding fix-point φ_T^*, the resulting assertion graph is denoted by G'.

Theorem 20. *Consider*

$$M \vDash_T G \Longleftrightarrow M \vDash_T G'. \tag{*}$$

Proof. According to Section 5.1 (the logic of algorithm), we know $(*)$ holds equivalent to the following $(**)$. For any $n \geq 2$, ρ_T represents all terminal paths of length n in assertion graph G (ρ_T is also the terminal path in assertion graph G') and σ represents all traces of the same length in model M. Consider

$$\sigma \vDash_{\operatorname{ant}} \rho_T \Longleftrightarrow \sigma \vDash_{\varphi_T^*} \rho_T. \tag{**}$$

The direction $\Longleftarrow$ is obvious, since $\varphi_T^*(e) \subseteq \operatorname{ant}(e)$ for any $e \in E$.

The direction $\Longrightarrow$: when $\sigma \vDash_{\operatorname{ant}} \rho_T$, according to Lemma 19,

$$\sigma[i] \in \varphi_T^*(\rho_T[i]) \quad (i = 1, 2, \ldots, n); \tag{39}$$

therefore, $\sigma \vDash_{\operatorname{ant}} \rho_T \Rightarrow \sigma \vDash_{\varphi_T^*} \rho_T$. □

G' can be obtained through a series of transformations on G. Theorem 20 guarantees that these transformation operations will not change the model's features on terminal satisfiability.

Theorem 21. *For any terminal assertion graph $G = (V, v_0, E, ant, cons, E_T)$ and any model M,*

$$TMC(M, G) \text{ returns true} \Longrightarrow M \vDash_T G. \tag{40}$$

Proof (G' is defined in Theorem 20). Consider

$$\begin{aligned}&\text{TMC returns true} \\ &\Longleftrightarrow \text{SMC}(M, G') \text{ returns true} \\ &\Longrightarrow M \vDash G' \quad (\text{SMC is complete [3, 8]}) \\ &\Longrightarrow M \vDash_T G' \quad (\text{Theorem 16}) \\ &\Longleftrightarrow M \vDash_T G \quad (\text{Theorem 20}).\end{aligned} \tag{41}$$

Theorem 21 proves that the TMC algorithm is sound. □

5.3. The TMC Algorithm Is Complete. Now let us prove the completeness of the algorithm. First, we need the following lemma.

Lemma 22 (see [3, 8]). *For any $n \geq 1$, $e \in E$, and any $s \in \psi_n(e)$, there is a finite initial path ρ and a finite trace σ of some length $l \leq n$ such that*

$$\rho[l] = e, \qquad \sigma[l] = s, \qquad \sigma \vDash_{ant} \rho. \tag{42}$$

Lemma 22 provides a counterexample for the proof of Theorem 25. We can find a path ρ and a trace σ satisfying $\sigma \vDash_{\operatorname{ant}} \rho$ but $\sigma \nvDash_{\operatorname{cons}} \rho$ when TMC returns false. $[\psi_1, \psi_2, \ldots, \psi_n, \ldots]$ is the simulation sequence used in SMC algorithm; for more, please read [3, 8].

Lemma 23 (see [3, 8]). *For all $n \geq 1$ and $e \in E$,*

$$\psi_n(e) \subseteq \psi_{n+1}(e), \qquad \psi_n(e) \subseteq ant(e). \tag{43}$$

According to the results of Lemma 23, we can conclude that $\psi_n(e) \subseteq \varphi_T^*(e)$ when $\operatorname{ant}(e) = \varphi_T^*(e)$.

Lemma 24. *For a terminal assertion graph $G = (V, v_0, E, ant, cons, E_T)$ and a model $M = (pre, post)$, let $[\varphi_1, \varphi_2, \varphi_3, \ldots]$ be the backward simulation sequence towards $E_T \subseteq E$. Then, for any $n \geq 1$, for any $e \in E$, and any state $s \in \varphi_n(e)$, there is a finite path ρ_e which starts from e and a finite trace σ_s which starts from s of some length l ($1 \leq l \leq n+1$) such that*

$$\rho_e[l] \in E_T, \qquad \sigma_s \vDash_{ant} \rho_e. \tag{44}$$

Proof (mathematical induction). k is a subscripted variable of the simulation sequence φ.

(1) ($k = 1$): when $e \in E_T$, we have $\varphi_1(e) = \operatorname{ant}(e)$; for all $s \in \varphi_1(e)$, there is a path $\rho_e = [e]$ of length 1 and a trace $\sigma_s = [s]$ such that

$$\rho_e[1] \in E_T, \qquad \sigma_s \vDash_{\operatorname{ant}} \rho_e. \tag{45}$$

When $e \in E - E_T$, we have two cases.

Case 1. $\operatorname{out}(e) \cap E_T = \emptyset$, $\varphi_1(e) = \emptyset$, the conclusion is obvious.

```
Input: the model $M$ and terminal assertion graph $G$.
Output: true: model terminally satisfies the terminal assertion graph.
        false: model doesn't satisfy the terminal assertion graph.
(1) repeat
(2) $G_{\text{old}} := G$;
(3) $\varphi_T^*$:= the fix-point of the backward simulation sequence towards $E_T$;
(4) $G$:= replace the antecedent function in $G$ with $\varphi_T^*$;
(5) until $G = G_{\text{old}}$;
(6) SMC$(M, G)$;
(7) end.
```

Algorithm 1: TMC(M, G).

Case 2. $\text{out}(e) \cap E_T \neq \emptyset$ and $\varphi_1(e) \neq \emptyset$; according to Definition 17:

$$\begin{aligned}&\forall s \in \varphi_1(e), \quad \exists e^+ \in \text{out}(e) \cap E_T,\\ &\text{s.t. } s \in \text{ant}(e) \cap \text{pre}(\text{ant}(e^+)) \Longrightarrow s \in \text{ant}(e), \qquad (46)\\ &s \in \text{pre}(\text{ant}(e^+));\end{aligned}$$

$$\overset{(46)}{\Longrightarrow} \exists s' \in \text{ant}(e^+), \quad \text{s.t. } s \in \text{pre}(s'), \qquad (47)$$

namely, there is a path $\rho_e = [ee^+]$ of length 2 and a trace $\sigma_s = [ss']$ such that

$$\rho_e[2] = e^+ \in E_T, \qquad \sigma_s \vDash_{\text{ant}} \rho_e. \qquad (48)$$

(2) Suppose the lemma is true for $k = n$ (n is an arbitrary integer), we can get for all $e \in E$, for all $s \in \varphi_n(e)$, there is a finite path ρ_e starting from e and a trace σ_s starting from s of some length l $(1 \leq l \leq n+1)$ such that

$$\rho_e[l] \in E_T, \qquad \sigma_s \vDash_{\text{ant}} \rho_e. \qquad (49)$$

(3) Suppose $k = n+1$ for all $e \in E$ and for all $s \in \varphi_{n+1}(e)$, according to Definition 17, we also have two cases.

Case 1. $s \in \varphi_n(e)$; the lemma is obviously true according to induction hypothesis.

Case 2. Consider

$$\begin{aligned}&\exists e^+ \in \text{out}(e), \quad \text{s.t. } s \in \text{pre}(\varphi_n(e^+)) \cap \text{ant}(e)\\ &\Longrightarrow s \in \text{ant}(e), \quad s \in \text{pre}(\varphi_n(e^+));\end{aligned} \qquad (50)$$

$$\overset{(50)}{\Rightarrow} \exists s' \in \varphi_n(e^+) \quad \text{s.t. } s \in \text{pre}(s') \qquad (51)$$

by the induction hypothesis, there exists a path ρ_{e^+} of length l $(1 \leq l \leq n+1)$ which starts from e^+ and a trace $\sigma_{s'}$ which starts from s' such that

$$\rho_{e^+}[l] \in E_T, \qquad \sigma_{s'} \vDash_{\text{ant}} \rho_{e^+}. \qquad (52)$$

According to (51), there is a path $\rho_e = (e : \rho_{e^+})$ of length $l+1$ and a trace $\sigma_s = (s : \sigma_{s'})$ of the same length such that

$$\rho_e[l+1] \in E_T, \qquad \sigma_s \vDash_{\text{ant}} \rho_e. \qquad (53)$$

(1), (2), and (3) show that the lemma is true. □

Lemma 24 ensures that the counterexample path which we find is the terminal path.

Theorem 25. *Consider*

$$TMC(M, G) \ \textit{returns false} \Longrightarrow M \nvDash_T G. \qquad (54)$$

Proof. Consider

$$\text{TMC}(M, G) \text{ returns false} \Longrightarrow \text{SMC}(M, G') \text{ returns false}; \qquad (55)$$

according to the SMC algorithm, there exists an edge $e_1 \in E$ such that $\psi^*(e_1) \nsubseteq \text{cons}(e_1)$. Let $s_1 \in \psi^*(e_1) - \text{cons}(e)$, according to the Lemma 22, there exists a finite path ρ_1 of length l_1 and a trace σ_1 of the same length such that

$$\begin{aligned}&\rho_1[l_1] = e_1, \qquad \sigma_1[l_1] = s_1,\\ &\sigma_1 \vDash_{\text{ant}} \rho_1, \quad \text{but } \sigma_1 \nvDash_{\text{cons}} \rho_1.\end{aligned} \qquad (56)$$

When $e_1 \in E_T$, ρ_1 is the terminal path. Namely, there exists an initial terminal path ρ_1 and a trace σ_1 of the same length such that $\sigma_1 \vDash_{\text{ant}} \rho_1$, $\sigma_1 \nvDash_{\text{cons}} \rho_1$; according to Definition 14, we get $M \nvDash_T G$.

When $e_1 \in E - E_T$, ρ_1 is not the terminal path, we know $s_1 \in \psi^*(e_1)$, $s_1 \notin \text{cons}(e_1)$. By analyzing Algorithm 1, we find that the terminal assertion graph G is already updated to G' when we call the SMC, and every edge in G' is labeled with $\varphi_T^*(e)/\text{cons}(e)$. In G', for all $e \in E$, $\text{ant}(e) = \varphi_T^*(e)$. According to Lemma 23, we get $s_1 \in \psi^*(e_1) \subseteq \varphi_T^*(e_1)$, $s_1 \notin \text{cons}(e_1)$. According to Lemma 24, there exists a finite path ρ_{e_1} of length l_2 which starts from e_1 and a trace σ_{s_1} of the same length which starts from s_1 such that

$$\rho_{e_1}[l_2] \in E_T, \qquad \sigma_{s_1} \vDash_{\text{ant}} \rho_{e_1}. \qquad (57)$$

Since $\rho_1[l_1] = \rho_{e_1}[1] = e_1$, ρ_1 and ρ_{e_1} can be spliced into one path ρ of length $l_1 + l_2 - 1$. Since $\sigma_1[l_1] = \sigma_{s_1}[1] = s_1$ is true, σ_1 and σ_{s_1} can be spliced into one trace σ whose length is $l_1 + l_2 - 1$. ρ is a terminal path and such that

$$\rho[l_1 + l_2 - 1] \in E_T, \qquad \sigma \vDash_{\text{ant}} \rho, \qquad \sigma \nvDash_{\text{cons}} \rho. \qquad (58)$$

According to Definition 14, we get $M \nvDash_T G$. □

Theorem 25 proves that the TMC algorithm is complete.

6. Conclusions

This paper has presented a theoretical and experimental study of the TMC process and related concepts, such as terminal assertion graph, terminal path, and terminal satisfiability. Under the basis of the concept mentioned above, to improve the efficiency of the method SMC and solve its limitation, this paper presented an algorithm TMC. The approach TMC is explained and discussed thoroughly in the body of the paper. Then, we use the hardware circuit round-robin arbiter to specify that TMC can be used in industry successfully. In the end, this paper proves that our approach is sound and complete. The method outlined here can be used to deal with finite specifications. It remains to be determined whether our approach will be suitable for infinite specifications.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

This work was supported by NSFC (nos. 60973016 and 61272175), 973 Program (no. 2010CB328004), and RFSGCC (no. SGSCDK00DWKJ1300510). The authors also would like to express their sincere thanks to Stephen Akobre and Lixiu Lin for useful discussions.

References

[1] E. M. Clarke, O. Grumberg, and D. A. Peled, *Model Checking*, MIT press, 1999.

[2] S. Hazelhurst and C.-J. H. Seger, "Simple theorem prover based on symbolic trajectory evaluation and BDD's," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 14, no. 4, pp. 413–422, 1995.

[3] S. Hazelhurst and C.-J. H. Seger, "Symbolic trajectory evaluation," in *Formal Hardware Verification*, pp. 3–78, Springer, New York, NY, USA, 1997.

[4] S. Owre, S. Rajan, J. M. Rushby, N. Shankar, and M. Srivas, "Pvs: combining specification, proof checking, and model checking," in *Computer Aided Verification*, pp. 411–414, Springer, 1996.

[5] C.-J. H. Seger and R. E. Bryant, "Formal verification by symbolic evaluation of partially-ordered trajectories," *Formal Methods in System Design*, vol. 6, no. 2, pp. 147–189, 1995.

[6] M. Pandey, R. Raimi, R. E. Bryant, and M. S. Abadir, "Formal verification of content addressable memories using symbolic trajectory evaluation," in *Proceedings of the 34th Design Automation Conference*, pp. 167–172, June 1997.

[7] M. Pandey and R. E. Bryant, "Exploiting symmetry when verifying transistor-level circuits by symbolic trajectory evaluation," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 18, no. 7, pp. 918–935, 1999.

[8] J. Yang and C.-J. H. Seger, "Introduction to generalized symbolic trajectory evaluation," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 11, no. 3, pp. 345–353, 2003.

[9] E. Clarke, O. Grumberg, S. Jha, Y. Lu, and H. Veith, "Progress on the state explosion problem in model checking," in *Informatics*, pp. 176–194, Springer, 2001.

[10] C.-T. Chou, "The mathematical foundation of symbolic trajectory evaluation," in *Computer Aided Verification*, pp. 196–207, Springer, 1999.

[11] C. Baier and J.-P. Katoen, *Principles of Model Checking*, MIT Press, Cambridge, Mass, USA, 2008.

[12] Y. Ramakrishna, C. Ramakrishnan, I. Ramakrishnan, S. A. Smolka, T. Swift, and D. S. Warren, "Efficient model checking using tabled resolution," in *Computer Aided Verification*, pp. 143–154, Springer, 1997.

[13] P. Gammie and R. Van Der Meyden, "Mck: model checking the logic of knowledge," in *Computer Aided Verification*, pp. 479–483, Springer, 2004.

[14] B. Bentley, "High level validation of next-generation microprocessors," in *Proceedings of the 7th IEEE International High-Level Design Validation and Test Workshop*, pp. 31–35, IEEE, 2002.

[15] A. Platzer, E. M. Clarke, and P. Zuliani, "Bayesian statistical model checking with application to Simulink/Stateflow verification," in *Proceedings of the 13th ACM international conference on Hybrid systems*, pp. 243–252, 2010.

[16] E. M. Clarke and P. Zuliani, "Statistical model checking for cyber-physical systems," in *Automated Technology for Verification and Analysis*, pp. 1–12, Springer, 2011.

[17] E. M. Clarke, O. Grumberg, and D. E. Long, "Model checking and abstraction," *ACM Transactions on Programming Languages and Systems*, vol. 16, no. 5, pp. 1512–1542, 1994.

[18] W. Visser, K. Havelund, G. Brat, S. Park, and F. Lerda, "Model checking programs," *Automated Software Engineering*, vol. 10, no. 2, pp. 203–232, 2003.

[19] A. J. Hu, J. Casas, and J. Yang, "Reasoning about gste assertion graphs," in *Correct Hardware Design and Verification Methods*, pp. 170–184, Springer, 2003.

[20] J. R. Burch, E. M. Clarke, K. L. McMillan, D. L. Dill, and L. J. Hwang, "Symbolic model checking: 10^{20} states and beyond," *Information and Computation*, vol. 98, no. 2, pp. 142–170, 1992.

[21] A. Biere, A. Cimatti, E. M. Clarke, O. Strichman, and Y. Zhu, "Bounded model checking," *Advances in Computers*, vol. 58, pp. 118–149, 2003.

[22] E. Clarke, A. Biere, R. Raimi, and Y. Zhu, "Bounded model checking using satisfiability solving," *Formal Methods in System Design*, vol. 19, no. 1, pp. 7–34, 2001.

[23] J. Yang and C. J. H. Seger, "Generalized symbolic trajectory evaluation," Intel SCL Tech. Rep, 2000.

[24] Y. Li, N. Zeng, W. N. N. Hung, and X. Song, "Enhanced symbolic simulation of a round-robin arbiter," in *Proceedings of the 29th IEEE International Conference on Computer Design (ICCD '11)*, pp. 102–107, November 2011.

[25] E. S. Shin, V. J. Mooney III, and G. F. Riley, "Round-robin arbiter design and generation," in *Proceedings of the 15th International Symposium on System Synthesis*, pp. 243–248, October 2002.

[26] Y. Li, S. Panwar, and H. Jonathan Chao, "On the performance of a Dual Round-Robin switch," in *Proceedings of the 20th Annual Joint Conference of the IEEE Computer and Communications Societies*, pp. 1688–1697, April 2001.

Hindawi Publishing Corporation
Journal of Applied Mathematics
Volume 2014, Article ID 702165, 13 pages
http://dx.doi.org/10.1155/2014/702165

Research Article

Counterexample-Preserving Reduction for Symbolic Model Checking

Wanwei Liu, Rui Wang, Xianjin Fu, Ji Wang, Wei Dong, and Xiaoguang Mao

School of Computer Science, National University of Defense Technology, Changsha 410073, China

Correspondence should be addressed to Wanwei Liu; wwliu@nudt.edu.cn

Received 12 February 2014; Accepted 14 April 2014; Published 14 May 2014

Academic Editor: Xiaoyu Song

The cost of LTL model checking is highly sensitive to the length of the formula under verification. We observe that, under some specific conditions, the input LTL formula can be reduced to an easier-to-handle one before model checking. In such reduction, these two formulae need not to be logically equivalent, but they share the same counterexample set w.r.t the model. In the case that the model is symbolically represented, the condition enabling such reduction can be detected with a lightweight effort (e.g., with SAT-solving). In this paper, we tentatively name such technique "counterexample-preserving reduction" (CePRe, for short), and the proposed technique is evaluated by conducting comparative experiments of BDD-based model checking, bounded model checking, and property directed reachability-(IC3) based model checking.

1. Introduction

LTL [1] is one of the most frequently used specification languages in model checking (cf. [2]). It designates properties over a linear structure, which can be viewed as an execution of the program. The task of LTL model checking is to search the state space (explicitly or implicitly), with the goal of detecting the existence of feasible traces violating the specification. If such traces exist, the model checker will report one of them as a "counterexample"; otherwise, the model checker will give an affirmative report.

It can be shown that the complexity of LTL model checking for $M \models \varphi$ is in $\mathcal{O}(|M| \times 2^{|\varphi|})$; meanwhile, the nesting depth of temporal operators might be the major factor affecting the cost in compiling LTL formulae. Hence, it is reasonable to simplify the specification before conducting model checking. For example, in [3], Somenzi and Bloem provided a set of rewriting schemas for simplifying LTL specifications, and these rewriting schemas preserve logical equivalence.

One may argue that "a majority of LTL formulae used in real applications are simple, succinct rather than complicated." Nevertheless, we might need to observe the following facts.

(i) Some LTL formula, for example $\mathbf{F}(p\mathbf{U}q)$, is usually considered to be a "simple" one. Nevertheless, it can be further simplified to $\mathbf{F}q$, and this fact tends to be ignored (on one hand, $p\mathbf{U}q$ implies $\mathbf{F}q$, and hence $\mathbf{F}(p\mathbf{U}q)$ implies $\mathbf{FF}q$ (i.e., $\mathbf{F}q$); on the other hand, q implies $p\mathbf{U}q$, and hence $\mathbf{F}q$ implies $\mathbf{F}(p\mathbf{U}q)$).

(ii) Indeed, people do use complicated specifications in the real industrial field, as well as in some standard benchmark (cf. [4]).

(iii) Last but not least, not all specifications are designated manually. Actually, some formulae are generated by specification-generaton-tools (e.g., ProSpec). Indeed, one may find that lots of these machine-generated specifications can be simplified.

Symbolic model checking [5] is one of the most significant breakthroughs in model checking, and two major fashions of symbolic model checking are widely used: one is the BDD-based manner [6] and the other is SAT-based manner, such as BMC [7] or PDR [8–11] algorithms.

Instead of using an explicit representation, the symbolic approach represents state space with a series of Boolean formulae. This enables implicit manipulation of the verification process and it usually leads to an efficient implementation [12]. Meanwhile, the symbolic encoding of transitions and invariants of the model provides heuristic information to simplify the specification. For example,

(i) the formulae $p\mathbf{U}q$ and $(r\mathbf{U}p)\mathbf{U}q$ can be, respectively, reduced as q and $(r\mathbf{U}p) \vee q$, if we know that $p \rightarrow q$ holds everywhere in the model;

(ii) each occurrence of $\mathbf{G}\theta$ in the specification can be replaced with $\top$ (i.e., logically true), if we can inductively infer that the Boolean formula θ holds at each reachable state in the model.

Actually, we can make sure of these conditions with the following efforts.

(i) To ensure that "$p \rightarrow q$ holds everywhere in the model", one possible way is to make sure that $p \rightarrow q$ is an *invariant* in the model—that is, just to examine if $\rho \wedge \neg(p \rightarrow q)$ is unsatisfiable (we in the latter denote it as $\rho \vdash p \rightarrow q$), where ρ is the Boolean encoding of the model's transition relation.

(ii) Likely, to justify that θ holds at each reachable state (note that a "dead-end" has no infinite path starting from it, and hence we may safely omit dead-ends in the model when doing this), it suffices to ensure that $\theta_0 \vdash \theta$ and $\rho \vdash \theta \rightarrow \theta'$, where θ_0 is the initial condition of the model.

We could do this because the component ρ should be satisfied at each transition step. Hence, it encloses both "local invariants" and "transitional invariants". For example, if $\rho = p \wedge (q \rightarrow q')$, then we may consider p as a local invariant, whereas $q \rightarrow q'$ is a transitional invariant.

Hence, this provides an opportunity to replace the specification with a simpler one, accompanied with some lightweight extra task of condition detection. Even if such detection fails, the overhead is usually negligible. More importantly, such reductions can be performed before starting model checking. At the same time, such kind of simplification is not always feasible to conduct manually; the reason is that the scale of boolean facilities is too large to be effectively handled by humans. Therefore, it is reasonable of leveraging the SAT-solvers to accomplish this task.

In this paper, we systematically investigate the above idea and tentatively name this technique *counter example-preserving reduction* (CePRe, for short). To justify it, we have extended NuSMV and implemented CePRe as an upfront option for LTL model checking. Subsequently, we conduct experiments over both industrial benchmarks and randomly generated cases. Experimental results show that CePRe can improve the efficiency significantly.

This paper is organized as follows. Section 2 revisits some basic notions. Section 3 introduces the details of the CePRe technique and Section 4 gives a formal treatment of performance analysis. In Section 5, the experimental results over industrial benchmarks and over random generated cases are given. We summarize the whole paper with Section 6.

2. Preliminaries

We presuppose a countable set $\mathscr{P}$ of *atomic propositions*, ranging over p, q, and so forth, and for each proposition $p \in \mathscr{P}$, we create a *primed version* p' (not belonging to $\mathscr{P}$) for it. For each set $\mathscr{V} \subseteq \mathscr{P}$, we define $\mathscr{V}' \triangleq \{p' \mid p \in \mathscr{V}\}$. We use $\mathbb{B}(\mathscr{V})$ to denote the set of Boolean formulae over $\mathscr{V}$. Similarly, we denote by $\mathbb{B}(\mathscr{V} \cup \mathscr{V}')$ the set of Boolean formulae built up from $\mathscr{V} \cup \mathscr{V}'$. The scope of the *prime* operator can be naturally lifted to Boolean formulae over $\mathbb{B}(\mathscr{V})$, by defining

$$\top' = \top, \qquad \bot' = \bot, \qquad (\neg\theta)' \triangleq \neg\theta', \\ (\theta_1 \longrightarrow \theta_2)' \triangleq \theta_1' \longrightarrow \theta_2'. \tag{1}$$

An *assignment* is a subset $\mathscr{V}$ of $\mathscr{P}$. Intuitively, it assigns 1 (or true) to the propositions belonging to $\mathscr{V}$ and assigns 0 (or false) to the other propositions. For each $\mathscr{V} \subseteq \mathscr{U} \subseteq \mathscr{P}$ and $\theta \in \mathbb{B}(\mathscr{U})$, we denote by $\mathscr{V} \Vdash \theta$ if θ is evaluated to 1 under the assignment $\mathscr{V}$.

A *united assignment* is a pair $(\mathscr{V}_1, \mathscr{V}_2)$, where both $\mathscr{V}_1$ and $\mathscr{V}_2$ are subsets of $\mathscr{P}$. It assigns 1 to the propositions belonging to $\mathscr{V}_1 \cup \mathscr{V}_2'$ and assigns 0 to the other propositions. Suppose that $\mathscr{V}_1, \mathscr{V}_2 \subseteq \mathscr{U} \subseteq \mathscr{P}$ and $\theta \in \mathbb{B}(\mathscr{U} \cup \mathscr{U}')$; we also write $(\mathscr{V}_1, \mathscr{V}_2) \Vdash \theta$ if θ is evaluated to 1 under the united assignment $(\mathscr{V}_1, \mathscr{V}_2)$.

LTL formulae can be inductively defined as follows.

(i) $\bot$ and $\top$ are LTL formulae.

(ii) Each proposition $p \in \mathscr{P}$ is an LTL formula.

(iii) If both φ_1 and φ_2 are LTL formulae, so does $\varphi_1 \rightarrow \varphi_2$.

(iv) If φ is an LTL formula, then $\mathbf{X}\varphi$ and $\mathbf{Y}\varphi$ are LTL formulae.

(v) If φ_1 and φ_2 are LTL formulae, then both $\varphi_1\mathbf{U}\varphi_2$ and $\varphi_1\mathbf{S}\varphi_2$ are LTL formulae.

Semantics of an LTL formula is defined w.r.t. a *linear structure* $\pi \in (2^{\mathscr{P}})^\omega$ (i.e., π is an infinite word over the alphabet $2^{\mathscr{P}}$) and a position $i \prec \omega$. Inductively:

(i) $\pi, i \models \top$ and $\pi, i \not\models \bot$;

(ii) $\pi, i \models p$ if and only if $\pi(i) \Vdash p$ (where $\pi(i)$ is the ith letter of π, which can be viewed as an assignment);

(iii) $\pi, i \models \varphi_1 \rightarrow \varphi_2$ if and only if either $\pi, i \not\models \varphi_1$ or $\pi, i \models \varphi_2$;

(iv) $\pi, i \models \mathbf{X}\varphi$ if and only if $\pi, i+1 \models \varphi$;

(v) $\pi, i \models \mathbf{Y}\varphi$ if and only if $i > 0$ and $\pi, i-1 \models \varphi$;

(vi) $\pi, i \models \varphi_1\mathbf{U}\varphi_2$ if and only if there is some $j \geq i$, s.t. $\pi, j \models \varphi_2$, and $\pi, k \models \varphi_1$ for each $i \leq k < j$;

(vii) $\pi, i \models \varphi_1\mathbf{S}\varphi_2$ if and only if there is some $j \leq i$, s.t. $\pi, j \models \varphi_2$ and $\pi, k \models \varphi_1$ for each $i \geq k > j$.

For the sake of convenience, we may directly write $\pi, 0 \models \varphi$ as $\pi \models \varphi$.

As usual, we employ some derived Boolean connectives such as

$$\neg\varphi \triangleq \varphi \longrightarrow \bot, \qquad \varphi \vee \psi \triangleq \neg\varphi \longrightarrow \psi, \\ \varphi \wedge \psi \triangleq \neg(\neg\varphi \vee \neg\psi) \tag{2}$$

and derived temporal operators such as

$$\mathbf{F}\varphi \triangleq \top\mathbf{U}\varphi, \qquad \mathbf{Z}\varphi \triangleq \neg\mathbf{Y}\neg\varphi, \qquad \mathbf{O}\varphi \triangleq \top\mathbf{S}\varphi,$$
$$\mathbf{G}\varphi \triangleq \neg\mathbf{F}\neg\varphi, \qquad \mathbf{H}\varphi \triangleq \neg\mathbf{O}\neg\varphi, \tag{3}$$
$$\varphi\mathbf{R}\psi \triangleq \neg\left(\neg\varphi\mathbf{U}\neg\psi\right), \qquad \varphi\mathbf{T}\psi \triangleq \neg\left(\neg\varphi\mathbf{S}\neg\psi\right).$$

Temporal operators like **X**, **U**, **F**, **G**, and **R** are called *future operators*, whereas **Y**, **Z**, **S**, **O**, **H,** and **T** are called *past operators*. An LTL formula is said to be *pure future* (resp., *pure past*) if it involves no past (resp., future) operators.

Theorem 1 (see [13]). *Each LTL formula has an equivalent pure future expression.*

Theorem 1 tells the fact that past operators do not add any expressive power to LTL formulae. Nevertheless, with these, we can give a much more succinct description in defining specifications.

Given an LTL formula φ, we denote by $\mathrm{sub}(\varphi)$ the set constituted with subformulae of φ. Particularly, we, respectively, denote by $\mathrm{sub}_{\mathbf{U}}(\varphi)$ and $\mathrm{sub}_{\mathbf{S}}(\varphi)$ the set consisting of "**U**-subformulae" and "**S**-subfomulae" of φ, where an **U**-formula (resp., **S**-formula) is a formula rooted at **U** (resp., **S**). (Note that $\mathbf{F}\varphi$ is also a **U**-formula whereas $\mathbf{G}\varphi$ is not.)

A *model* is a tuple $M = \langle \mathscr{V}, \rho, \theta_0, \mathscr{F}\rangle$, where

(i) $\mathscr{V} \subseteq \mathscr{P}$ is a finite set of atomic propositions,

(ii) $\rho \in \mathbb{B}(\mathscr{V} \cup \mathscr{V}')$ is the *transition relation*,

(iii) $\theta_0 \in \mathbb{B}(\mathscr{V})$ is the *initial condition*,

(iv) $\mathscr{F} \subseteq \mathbb{B}(\mathscr{V})$ is a set of *fairness constraints*.

A *derived linear structure* of M is an infinite word $\pi \in (2^{\mathscr{V}})^{\omega}$, such that

(1) $\pi(0) \Vdash \theta_0$;

(2) $(\pi(i), \pi(i+1)) \Vdash \rho$ for each $i \prec \omega$;

(3) for each $\varphi \in \mathscr{F}$, there are infinitely many i's having $\pi(i) \Vdash \varphi$.

We denote by $\mathbf{L}(M)$ the set of derived linear structures of M and call it the *language* of M.

For a model M and an LTL formula φ, we denote as $M \vDash \varphi$ if $\pi \vDash \varphi$ for each $\pi \in \mathbf{L}(M)$. Meanwhile, we define

$$\mathbf{CE}(\varphi, M) \triangleq \{\pi \in \mathbf{L}(M) \mid \pi \nvDash \varphi\} \tag{4}$$

and call it the *counterexample set* of φ w.r.t. M.

3. Counterexample-Preserving Reduction

We describe the CePRe technique in this section, but first of all, let us fix the components of the model and just let M be $\langle \mathscr{V}, \rho, \theta_0, \mathscr{F}\rangle$ in the rest of this section. For M, we are particularly concerned about formulae having the same counterexample set—we say that φ and ψ are *interreduceable* w.r.t. M if and only if $\mathbf{CE}(\varphi, M) = \mathbf{CE}(\psi, M)$, denoted as $\varphi \approx_M \psi$. Hence, $\varphi \approx_M \psi$ implies that $M \vDash \varphi \Leftrightarrow M \vDash \psi$.

Table 1: Reduction rules about model components.

$\theta_0 \vdash \theta \rhd \theta \approx \top$ (Init)	$\rho \vdash \theta \rhd \mathbf{G}\theta \approx \top$ (Trans)
$\theta \in \mathscr{F} \rhd \mathbf{GF}\theta \approx \top$ (Fair)	$\theta_0 \vdash \theta; \rho \vdash \theta \to \theta' \rhd \mathbf{G}\theta \approx \top$ (Ind)

Table 2: Reduction rules of (Conj) and (Disj).

$(\mathbf{P}^w\varphi \wedge \mathbf{P}^s\varphi) \approx \mathbf{P}^s\varphi$ (Conj)	$(\mathbf{P}^w\varphi \vee \mathbf{P}^s\varphi) \approx \mathbf{P}^w\varphi$ (Disj)

Table 3: Reduction rules for formulae involving nested pure future operators.

$\mathbf{F}(\varphi\mathbf{U}\psi) \approx \mathbf{F}\psi$ (FU)	$\varphi\mathbf{U}(\mathbf{F}\psi) \approx \mathbf{F}\psi$ (U_{F})
$\mathbf{FF}\varphi \approx \mathbf{F}\varphi$ (FF)	$\mathbf{GFG}\varphi \approx \mathbf{FG}\varphi$ (GFG)

The central part of CePRe is a series of *reduction rules* being of the form

$$\text{Cond} \rhd \varphi \approx_M \psi\,(\text{NAME}), \tag{5}$$

where "Cond" is called the *additional condition*.

Though the relation $\approx_M$ is actually symmetric, we always write the reduced formula on the righthand of the "$\approx$" sign in a reduction rule. Since the model M is fixed, in this section, we omit it from the subscript. In addition, if the additional condition trivially holds, we will discard this part and directly write the rule as $\varphi \approx \psi$, and in this case we say that this rule is "*model-independent*"; otherwise, we say that the underlying reduction rule is "*model-dependent.*"

3.1. The Basic Reduction Rules. In this section, we define some basic CePRe rules. First of all, we have some elementary reduction rules relating to model components, as depicted in Table 1. For the rules (Init), (Ind), and (Trans), the notation "$\vdash$" occurring in the condition part stands for the "*inferring*" relation in propositional logic ($\rho \vdash \theta$ if and only if $\rho \wedge \neg\theta$ is unsatisfiable), and we here require that $\theta, \theta_1, \theta_2 \in \mathbb{B}(\mathscr{V})$.

Subsequently, let us define a partial order "$\sqsubseteq$" over unary temporal operators (and their combinations) as follows:

$$\mathbf{F} \sqsubseteq \mathbf{GF} \sqsubseteq \mathbf{FG} \sqsubseteq \mathbf{G}$$
$$\mathbf{F} \sqsubseteq \mathbf{X}^i \sqsubseteq \mathbf{G} \quad (i \prec \omega) \tag{6}$$
$$\mathbf{O} \sqsubseteq \mathbf{HO} \sqsubseteq \mathbf{OH} \sqsubseteq \mathbf{H},$$

where $\mathbf{X}^0\varphi \triangleq \varphi$ and $\mathbf{X}^{i+1}\varphi \triangleq \mathbf{X}(\mathbf{X}^i\varphi)$.

Assume that $\mathbf{P}^w, \mathbf{P}^s \in \{\mathbf{F}, \mathbf{FG}, \mathbf{GF}, \mathbf{G}, \mathbf{O}, \mathbf{HO}, \mathbf{OH}, \mathbf{H}\} \cup \{\mathbf{X}^i \mid i \prec \omega\}$ and $\mathbf{P}^w \sqsubseteq \mathbf{P}^s$; then we have two model-indenpendent rules, as depicted in Table 2. Though these rules seem to be trivial, they are useful in doing combinational reductions (see the example given in Section 3.3).

Table 3 provides some reduction rules that can be used to simplify nested temporal operators.

Since we always stand at the starting point when doing model checking (i.e., the goal is to check if $\pi, 0 \vDash \varphi$ for each $\pi \in \mathbf{L}(M)$), we can sometimes "erase" the outermost past operators, as depicted in Table 4.

Table 5 introduces a series of rules handling formulae involving adjacent past and future temporal operators.

Table 4: Reduction rules for formulae involving (outermost) past operators.

$\mathbf{Y}\varphi \approx \bot$ (Y)	$\mathbf{O}\varphi \approx \varphi$ (O)	$\varphi\mathbf{S}\psi \approx \varphi$ (S)

Table 5: Reduction rules for formulae involving adjacent past and future operators.

$\mathbf{XY}\varphi \approx \varphi$ (XY)	$\mathbf{FH}\varphi \approx \mathbf{H}\varphi$ (FH)
$\mathbf{FO}\varphi \approx \mathbf{F}\varphi \vee \mathbf{O}\varphi$ (FO)	$\mathbf{F}(\varphi\mathbf{S}\psi) \approx \mathbf{F}\psi \vee \varphi\mathbf{S}\psi$ (FS)

We let $\theta_1, \theta_2, \dots$ range over $\mathbb{B}(\mathscr{V})$ and let $\varphi_1, \varphi_2, \dots$ be arbitrary LTL formulae, and then we have some model-dependent rules. The first group of such rules is listed in Table 6. Table 7 provides another set of model-dependent reduction rules, and these rules are mainly concerned with LTL formulae involving adjacent **U**-operators. Lastly, Table 8 provides some reduction rules that can be used to simplify formulae with mixed usage of **U** and **R**.

3.2. Principles for Rule Generation. Actually, we may acquire more CePRe rules by applying some *principles* to the existing rules. In this section, we introduce four major principles in our framework.

3.2.1. The Inversion Principle. We say that the temporal opertors "**U** and **S**," "**R** and **T**," "**G** and **H**," and "**F** and **O**" are pairwise *inverse operators* (The inverse operator of **X** may be **Z** or **Y**, which is judged from the context. Fortunately, in this paper, we need not consider this case). Then, for CePRe rules list in Table 3 and Table 6–Table 8, we may apply the following principle:

$$\frac{\rho \vdash \theta \rhd \varphi \approx \psi}{\rho \vdash \widetilde{\theta} \rhd \mathsf{inv}(\varphi) \approx \mathsf{inv}(\psi)}, \tag{7}$$

namely, *Inversion Principle*. In detail:

(i) if $\theta \in \mathbb{B}(\mathscr{V})$ then $\widetilde{\theta} = \theta$; if $\theta \in \mathbb{B}(\mathscr{V} \cup \mathscr{V}')$, then $\widetilde{\theta}$ is obtained from θ by switching the primed and unprimed propositions;

(ii) $\mathsf{inv}(\varphi)$ and $\mathsf{inv}(\psi)$ are obtained from φ and ψ by swithing the temporal operators with their inverse operators, respectively.

For example, one can apply this principle to (U) and (R), and then the following two rules

$$\begin{gathered} \rho \vdash \theta_1 \vee \theta_2 \rhd \theta_1\mathbf{S}\theta_2 \approx \mathbf{O}\theta_2 \quad (\mathrm{S}) \\ \rho \vdash \theta_2' \longrightarrow \theta_1' \vee \theta_2 \rhd \theta_1\mathbf{T}\theta_2 \approx \theta_2 \quad (\mathrm{T}) \end{gathered} \tag{8}$$

are immediately obtained.

3.2.2. The Duality Principle. We say that "$\top$ and $\bot$," "$\wedge$ and $\vee$," "**F** and **G**," "**O** and **H**," "**Y** and **Z**," "**X** and **X** itself," "**U** and **R**," and "**T** and **S**" are pairwise the *dual operators*. Then, the *duality principle* could be described as follows:

$$\frac{\mathrm{Cond} \rhd \varphi \approx \psi}{\overline{\mathrm{Cond}} \rhd \mathsf{dual}(\varphi) \approx \mathsf{dual}(\psi)}. \tag{9}$$

Table 6: Reduction rules of (U) and (R).

$\rho \vdash \theta_1 \vee \theta_2 \rhd \theta_1\mathbf{U}\theta_2 \approx \mathbf{F}\theta_2$ (U)
$\rho \vdash \theta_2 \rightarrow \theta_1 \vee \theta_2' \rhd \theta_1\mathbf{R}\theta_2 \approx \theta_2$ (R)

Table 7: Reduction rules for formulae involving adjacent **U** operators.

$\rho \vdash \theta_1 \rightarrow \theta_2 \rhd \theta_1\mathbf{U}\theta_2 \approx \theta_2$	$(\mathrm{U}[1 \rightarrow 2])$
$\rho \vdash \theta_1 \rightarrow \theta_3 \rhd (\theta_1\mathbf{U}\varphi_2)\mathbf{U}\theta_3 \approx \varphi_2\mathbf{U}\theta_3$	$(\mathrm{U}^{\mathrm{U}}[1 \rightarrow 3])$
$\rho \vdash \theta_2 \rightarrow \theta_3 \rhd (\varphi_1\mathbf{U}\theta_2)\mathbf{U}\theta_3 \approx \theta_3 \vee (\varphi_1\mathbf{U}\theta_2)$	$(\mathrm{U}^{\mathrm{U}}[2 \rightarrow 3])$
$\rho \vdash \theta_3 \rightarrow \theta_2 \rhd (\varphi_1\mathbf{U}\theta_2)\mathbf{U}\theta_3 \approx (\varphi_1 \vee \theta_2)\mathbf{U}\theta_3$	$(\mathrm{U}^{\mathrm{U}}[3 \rightarrow 2])$
$\rho \vdash \theta_2 \rightarrow \theta_3' \rhd (\varphi_1\mathbf{U}\theta_2)\mathbf{U}\theta_3 \approx (\varphi_1 \vee \theta_2)\mathbf{U}\theta_3$	$(\mathrm{U}^{\mathrm{U}}[2 \rightarrow 3'])$
$\rho \vdash \neg\theta_2 \rightarrow \theta_3 \rhd (\varphi_1\mathbf{U}\theta_2)\mathbf{U}\theta_3 \approx \mathbf{F}\theta_3$	$(\mathrm{U}^{\mathrm{U}}[\neg 2 \rightarrow 3])$
$\rho \vdash \theta_1 \rightarrow \theta_2 \rhd \theta_1\mathbf{U}(\theta_2\mathbf{U}\varphi_3) \approx \theta_2\mathbf{U}\varphi_3$	$(\mathrm{U}_{\mathrm{U}}[1 \rightarrow 2])$
$\rho \vdash \theta_1 \rightarrow \theta_3 \rhd \theta_1\mathbf{U}(\varphi_2\mathbf{U}\theta_3) \approx \varphi_2\mathbf{U}\theta_3$	$(\mathrm{U}_{\mathrm{U}}[1 \rightarrow 3])$
$\rho \vdash \theta_2 \rightarrow \theta_1 \rhd \theta_1\mathbf{U}(\theta_2\mathbf{U}\varphi_3) \approx \theta_1\mathbf{U}\varphi_3$	$(\mathrm{U}_{\mathrm{U}}[2 \rightarrow 1])$

Table 8: Reduction rules for formulae involving adjacent **U** and **R** operators.

$\rho \vdash \theta_1 \rightarrow \theta_3 \rhd (\theta_1\mathbf{R}\varphi_2)\mathbf{U}\theta_3 \approx ((\theta_1\mathbf{R}\varphi_2) \vee \theta_3) \wedge \mathbf{F}\theta_3$	$(\mathrm{U}^{\mathrm{R}}\ [1 \rightarrow 3])$
$\rho \vdash \neg\theta_1 \rightarrow \theta_3 \rhd (\theta_1\mathbf{R}\varphi_2)\mathbf{U}\theta_3 \approx \varphi_2\mathbf{U}\theta_3$	$(\mathrm{U}^{\mathrm{R}}\ [\neg 1 \rightarrow 3])$
$\rho \vdash \theta_1 \rightarrow \theta_3 \rhd \theta_1\mathbf{U}(\varphi_2\mathbf{R}\theta_3) \approx \varphi_2\mathbf{R}\theta_3$	$(\mathrm{U}_{\mathrm{R}}\ [1 \rightarrow 3])$

We require that the condition "Cond" should be either trivial (in this case, the rule is a model-independent one) or of the form $\rho \vdash \theta_1 \rightarrow \theta_2$, and in the latter case we have $\overline{\mathrm{Cond}} = \rho \vdash \theta_2 \rightarrow \theta_1$. The formulae $\mathsf{dual}(\varphi)$ and $\mathsf{dual}(\psi)$ are obtained from φ and ψ by switching the operators with their dual.

For example, apply the duality principle to the rules (FU), $(\mathrm{U_F})$, (FF), and (GFG), one may get the following new rules:

$$\begin{gathered} \mathbf{G}(\varphi\mathbf{R}\psi) \approx \mathbf{G}\psi \quad (\mathrm{GR}), \qquad \varphi\mathbf{R}(\mathbf{G}\psi) \approx \mathbf{G}\psi \quad (\mathrm{R_G}), \\ \mathbf{GG}\varphi \approx \mathbf{G}\varphi \quad (\mathrm{GG}), \qquad \mathbf{FGF}\varphi \approx \mathbf{GF}\varphi \quad (\mathrm{FGF}). \end{gathered} \tag{10}$$

Note that when applying the duality principle to model-dependent rules, besides switching the operators, we also need to exchange the antecedent and subsequent on the righthand of $\vdash$ in the condition part, in the case that the condition is of the form $\rho \vdash \theta_1 \rightarrow \theta_2$. As an example, we may obtain the reduction rule

$$\rho \vdash \theta_3 \longrightarrow \theta_2 \rhd (\varphi_1\mathbf{R}\theta_2)\mathbf{R}\theta_3 \approx \theta_3 \wedge (\varphi_1\mathbf{R}\theta_2) \quad \left(\mathrm{R}^{\mathrm{R}}[3 \longrightarrow 2]\right), \tag{11}$$

by applying the duality principle to $(\mathrm{U}^{\mathrm{U}}[2 \rightarrow 3])$.

3.2.3. The Composition Principle. The *composition principle* can be formally described as follows:

$$\frac{\begin{gathered}\mathrm{Cond}_1 \rhd \varphi_1 \approx \psi_1 \\ \mathrm{Cond}_2 \rhd \varphi_2 \approx \psi_2\end{gathered}}{\mathrm{Cond}_1\ \&\ \mathrm{Cond}_2 \rhd \varphi_1 \approx {\psi_1}^{\varphi_2}_{\psi_2}}; \tag{12}$$

that is, if φ_i and ψ_i are interreduceable under the conditin Cond_i $(i = 1, 2)$, then φ_1 and ${\psi_1}^{\varphi_2}_{\psi_2}$ are also interreduceable. In using this principle, we have the following constraints.

```
    Input: The original specification $\varphi$.
    Output: The reduced specification.
(1) let $\Gamma := \emptyset$;  /* $\Gamma$ memorizes the sub-formulae with infeasible condition */
(2) let $\Delta := \{\psi \in (\mathrm{sub}(\varphi) \backslash \Gamma)$ such that $\psi$ matches some reduction rule(s)$\}$;
(3) foreach $\psi_1, \psi_2 \in \Delta$ s.t. $\psi_1 \neq \psi_2$ do
(4)     if $\psi_1 \in \mathrm{sub}(\psi_2)$ then
(5)         $\Delta := \Delta \backslash \{\psi_1\}$;  /* that is, we only proceed "max" subformulae */
(6)     end
(7) end
(8) if $\Delta = \emptyset$ then
(9)     return $\varphi$;
(10) end
(11) foreach $\psi \in \Delta$ do
(12)     let $\Theta :=$ the set of rules that can be applied to $\psi$;
(13)         /* note that we have $|\Theta| \leq 5$ for each $\psi$ */
(14)     while $\Theta \neq \emptyset$ do
(15)         choose $R := (\mathrm{Cond} \rhd \psi \approx \eta)$ in $\Theta$;
(16)         if Cond is stated then
(17)             $\varphi := \varphi^{\psi}_{\eta}$;  /* $\varphi^{\psi}_{\eta}$ is obtained from $\varphi$ by replacing $\psi$ with $\eta$ */
(18)             break;
(19)         end
(20)         $\Theta := \Theta \backslash \{R\}$;
(21)     end
(22)     $\Delta := \Delta \backslash \{\psi\}$;
(23)     if $\Theta = \emptyset$ then
(24)         $\Gamma := \Gamma \cup \{\psi\}$; /* $\psi$ would be excluded in the next iteration */
(25)     end
(26) end
(27) goto 2;
```

Algorithm 1: The "max-match" rule-selection strategy.

(1) Cond_2 must be θ_0-free; that is, it must be irrelevant to the initial condition.

(2) If the second rule is one of these listed in Table 4 (or the inversion/duality version), then we require that the (designated) occurrences of φ_2 in ψ_1 must be *temporally outermost*; that is, φ_2 does not appear in the scope of any temporal operators.

(3) Since $\eta \approx \eta$ trivially holds, we have the following special case of the composition principle:

$$\frac{\mathrm{Cond} \rhd \varphi \approx \psi}{\mathrm{Cond} \rhd \eta \approx \eta^{\varphi}_{\psi}}, \tag{13}$$

and call it the local application of (the above) CePRe rule.

To explaint the reason why we need to impose the second constraint, just consider the following fact: we have $\mathbf{Y}\varphi \approx \bot$ according to (Y); yet this does not imply that $\mathbf{FY}\varphi \approx \mathbf{F} \bot$ holds.

3.2.4. The Decomposition Principle. The last principle, which could be corporately used in *modular model checking* [14], is described as follows:

$$\frac{\begin{array}{c}\mathrm{Cond}_1 \rhd \varphi_1 \approx_{M_1} \psi_1 \\ \mathrm{Cond}_2 \rhd \varphi_2 \approx_{M_2} \psi_2\end{array}}{\mathrm{Cond}_1 \ \& \ \mathrm{Cond}_2 \rhd \varphi_1 \approx_{M_1 \| M_2} \psi_1{}^{\varphi_2}_{\psi_2}}, \tag{14}$$

where all constraints imposed to the *composition principle* are still required. $M_1 \| M_2$ is the *synchronized composition* of M_1 and M_2; that is, $M_1 \| M_2 = \langle \mathscr{V}_1 \cup \mathscr{V}_2, \rho_1 \wedge \rho_2, \theta_{0,1} \wedge \theta_{0,2}, \mathscr{F}_1 \cup \mathscr{F}_2 \rangle$ if $M_i = \langle \mathscr{V}_i, \rho_i, \theta_{0,i}, \mathscr{F}_i \rangle$.

3.3. Reduction Strategy. We show the usage of CePRe reduction rules by illustrating the reduction process of $M \vDash (\theta_1 \mathbf{U} \theta_2) \mathbf{U} \theta_3$; see Algorithm 1.

(1) We may first try with the rule $(\mathrm{U}^{\mathrm{U}}[1 \rightarrow 3])$ by inquiring the SAT-solver if $\rho \vdash \theta_1 \rightarrow \theta_3$ holds.

(2) If the SAT-solver returns "unsatisfiable" with the input $\rho \wedge \theta_1 \wedge \neg\theta_3$, then it implies that the additional condition is stated, and we may replace the specification with $\theta_2 \mathbf{U} \theta_3$.

(3) Otherwise, we will try with another reduction rule, such as $(\mathrm{U}^{\mathrm{U}}[2 \rightarrow 3])$.

Compositional use of reduction rules may lead to a more aggressive reduction. For example,

(1) for the task of model checking $M \vDash \mathbf{FO}p$, we may firstly change the goal as $M \vDash \mathbf{F}p \vee \mathbf{O}p$, according to the rule (FO);

(2) now, the subformula $\mathbf{O}p$ is a temporally outermost one; hence we may make a local application of (O), and then the goal becomes $M \vDash \mathbf{F}p \vee p$;

(3) finally, we may change the model checking problem into $M \models \mathbf{F}p$ via the rule (Disj).

In the real implementation, we perform a "*max-match*" rule-selection strategy, as depicted in Algorithm 1. Note that in this algorithm, only

(1) basic rules,

(2) rules obtained from the inversion principle or the duality principle

could be directly used (maybe in a local manner). This guarantees the finiteness for rule selection.

In line 15, a rule having simpler condition always has the higher precedence to be chosen. Hence, a model-independent rule always has a higher priority than a model-dependent one.

Lastly, we can see that the reduction can be accomplished within $\mathcal{O}(|\varphi|)$ iterations.

4. Performance Analysis of CePRe

We now try to answer the question "why we can gain a better performance during verification if CePRe is conducted first." To give a rigorous explanation, we briefly revisit the implementation of some symbolic model checking algorithms, and we are mainly concerned about the following techniques:

(1) the BDD-based model checking technique,

(2) the bounded model checking technique (BMC), for both syntactic encoding and semantic encoding approaches,

(3) the property directed reachability (alternatively, IC3) algorithm.

4.1. Analysis on BDD-Based Model Checking.

The core procedure of BDD-based LTL symbolic model checking algorithm is to construct a *tableau* for the (negated) property. In the following, we refer the tableau of $\neg\varphi$ as $T_{\neg\varphi}$, and we would give an analysis on its major components affecting the cost of model checking.

State Space. The state space of $T_{\neg\varphi}$ consists of subsets of $el(\varphi)$, and the set $el(\varphi)$ can be inductively computed as follows:

(i) $el(\top) = el(\bot) = \emptyset$;

(ii) $el(p) = \{p\}$ if $p \in \mathcal{P}$;

(iii) $el(\varphi_1 \rightarrow \varphi_2) = el(\varphi_1) \cup el(\varphi_2)$;

(iv) $el(\mathbf{X}\psi) = \{\mathbf{X}\psi\} \cup el(\psi)$, and $el(\mathbf{Y}\psi) = \{\mathbf{Y}\psi\} \cup el(\psi)$;

(v) $el(\varphi_1 \mathbf{U} \varphi_2) = el(\varphi_1) \cup el(\varphi_2) \cup \{\mathbf{X}(\varphi_1 \mathbf{U} \varphi_2)\}$ and $el(\varphi_1 \mathbf{S} \varphi_2) = el(\varphi_1) \cup el(\varphi_2) \cup \{\mathbf{Y}(\varphi_1 \mathbf{S} \varphi_2)\}$.

We can see from the definition that $el(\varphi) = el(\neg\varphi)$ holds. With symbolic representation, each formula $\psi \in el(\varphi)$ corresponds to a proposition in building the tableau. Moreover, if $\psi \in \mathcal{P}$, then no new proposition needs to be introduced (since it has already been introduced in building the symbolic representation of M); otherwise, a fresh proposition p_ψ is required. Hence the total number of newly introduced propositions equals $|el(\varphi) \setminus \mathcal{P}|$. From an induction over formula's structure, we have the following claim.

Proposition 2. *$|el(\varphi) \setminus \mathcal{P}|$ equals the number of temporal operators in φ.*

Transitions. The transition relation of $T_{\neg\varphi}$ is a conjunction of a set of constraints, and each constraint is either of the form $p_{\mathbf{X}\psi} \leftrightarrow (\sigma(\psi))'$ or $p'_{\mathbf{Y}\eta} \leftrightarrow \sigma(\eta)$, where $\mathbf{X}\psi, \mathbf{Y}\eta \in el(\varphi)$, and the function σ can be inductively defined as follows:

(i) $\sigma(\bot) = \bot$ and $\sigma(\top) = \top$;

(ii) $\sigma(p) = p$ for each $p \in \mathcal{P}$;

(iii) $\sigma(\psi_1 \rightarrow \psi_2) = \sigma(\psi_1) \rightarrow \sigma(\psi_2)$;

(iv) $\sigma(\mathbf{X}\psi_1) = p_{\mathbf{X}\psi_1}$ and $\sigma(\mathbf{Y}\psi_2) = p_{\mathbf{Y}\psi_2}$;

(v) $\sigma(\psi_1 \mathbf{U} \psi_2) = \sigma(\psi_2) \vee \sigma(\psi_1) \wedge p_{\mathbf{X}(\psi_1 \mathbf{U} \psi_2)}$ and $\sigma(\psi_1 \mathbf{S} \psi_2) = \sigma(\psi_2) \vee \sigma(\psi_1) \wedge p_{\mathbf{Y}(\psi_1 \mathbf{U} \psi_2)}$.

According to the definition of el, we can see that each $\psi \in \mathrm{sub}(\varphi)$ rooted at a future (resp., past) temporal operator exactly produces one formula $\mathbf{X}\eta$ (resp., $\mathbf{Y}\eta$) in $el(\varphi)$, and hence a new proposition $p_{\mathbf{X}\eta}$ (resp., $p_{\mathbf{Y}\eta}$) would be introduced. Subsequently, each such $p_{\mathbf{X}\eta}$ (resp., $p_{\mathbf{Y}\eta}$) adds exactly one constraint to the transition relation. Hence, we have the following claim.

Proposition 3. *The number of constraints in the transition relation of $T_{\neg\varphi}$ equals the number of temporal operators occurring in φ (alternatively, $|el(\varphi) \setminus \mathcal{P}|$).*

Fairness Constraints. According to the tableau construction, each $\psi \in \mathrm{sub}_{\mathbf{U}}(\neg\varphi)$ would impose a fairness constraint to $T_{\neg\varphi}$. Hence, the number of fairness constraints equals $|\mathrm{sub}_{\mathbf{U}}(\neg\varphi)|$.

With a case-by-case checking, we can show the following theorem.

Theorem 4. *Let "$Cond \rhd \varphi \approx \psi$" be a reduction rule; then we have $|el(\psi) \setminus \mathcal{P}| \leq |el(\varphi) \setminus \mathcal{P}|$ and $|sub_{\mathbf{U}}(\psi)| \leq |sub_{\mathbf{U}}(\neg\varphi)|$.*

4.2. Analysis on Bounded Model Checking.

In contrast, the cost of BMC is quite sensitive to the encoding approach. In a broad sense, we can categorize the encoding approaches into two fashions.

Syntactic Encodings. Such kind of encodings are inductively produced w.r.t. the formula's structure. The very original one is presented in [7], and this is improved in [15] by observing some properties of that encoding. In [16, 17], a linear incremental syntactic encoding is suggested, and see [18] for the translation for ECTL*.

Semantic Encodings. In [19], an alternative BMC technique is provided: it mimics the tableau-based model checking process, but it expresses the fair-path detection upon the product model with Boolean formula. (In [20], a "fixpoint"-based encoding is proposed, and it can also be subsumed to semantic encodings.)

For the semantic encodings, the reason that we can benefit from CePRe is exactly the same as that for BDD-based approach. Because, the encoding is a conjunction of a k-step unrolling of M and a k-step unrolling of $T_{\neg\varphi}$ (an unrolling is either a partial derived linear structure or one ending with a lasso). The former is usually in a fixed pattern, and for the latter, we need $k\times|el(\varphi)\setminus\mathscr{P}|$ new propositions, and the sizes of Boolean formulae w.r.t the transition and fairness constraints (note that the part w.r.t. fairness constraints can be linearized) are, respectively, $\mathcal{O}(k\times|el(\varphi)\setminus\mathscr{P}|)$ and $\mathcal{O}(k^2\times|\mathrm{sub}_{\mathbf{U}}(\neg\varphi)|)$.

For a syntactic BMC encoding, one needs to generate a Boolean formula of the form $E^k_M \wedge E^k_{\neg\varphi}$, where E^k_M is the unrolling of M with k steps, and $E^k_{\neg\varphi}$ describes that such k-step unrolling would cause a violation of φ. In general, E^k_M is almost the same in all kinds of syntactic encodings, and the key factor affecting the cost lies in $E^k_{\neg\varphi}$.

Given a subformula ψ of φ, if we use $||E^k_\psi||$ to denote the maximum length of the Boolean formula describing that ψ is initially satisfied upon a k-step unrolling, then it can be inductively computed as follows:

(i) $||E^k_\bot|| = ||E^k_\top|| = 0$; (this is just for the case when $\bot$ or $\top$ appears as a subformula in the specification and hence can be optimized; otherwise, we have $||E^k_\bot|| = ||E^k_\top|| = 1$);

(ii) $||E^k_p|| = 1$ for each $p\in\mathscr{P}$;

(iii) $||E^k_{\varphi_1\to\varphi_2}|| = ||E^k_{\varphi_1}|| + ||E^k_{\varphi_2}|| + 1$;

(iv) $||E^k_{\mathbf{X}\psi}|| = ||E^k_{\mathbf{Y}\psi}|| = ||E^k_\psi||$;

(v) $||E^k_{\varphi_1\mathbf{U}\varphi_2}|| = ||E^k_{\varphi_1\mathbf{S}\varphi_2}|| = L(k)\times||E^k_{\varphi_1}||+k\times||E^k_{\varphi_2}||$. (Note that this case does not imply that further blowup would be caused with deeper nesting of temporal operators. For example, in [17], by introducing fresh propositions and reusing, it still leads to a linear encoding for the whole formula.)

Here, $L(k)$ is some polynomial about k, related to the encoding approach. For example, with the technique proposed in [7, 15], we have $L(k)\in\mathcal{O}(k^2)$, whereas $L(k)\in\mathcal{O}(k)$ in [16, 17]. This partly explains the reason that we tend to change temporal nestifications with Boolean combinations, as done in ($\mathrm{U}^{\mathrm{U}}[3\to 2]$) and so forth.

Another feature affecting the cost is the scale of propositions required for the encoding. If we denote by $\mathrm{var}_k(\varphi)$ the set of additional propositions which only takes part in the encoding of $E^k_{\neg\varphi}$, then we have the following conclusions.

(i) For the techniques proposed in [7, 15], we have $\mathrm{var}_k(\varphi) = 0$. i.o.w.; all propositions required in encoding $E^k_\neg\varphi$ can be shared with those for E^k_M.

(ii) In term of the encoding presented in [17], we need to add $\mathcal{O}(k)$ new propositions to $\mathrm{var}_k(\varphi)$ for each $\mathbf{U}$-subformula and for each $\mathbf{S}$-subformula.

Theorem 5. *Let "Cond $\rhd\ \varphi\approx\psi$" be a reduction rule; then we have $||E^k_\psi||\le||E^k_\varphi||$ and $|var_k(\psi)|\le|var_k(\varphi)|$ in syntactic encodings.*

4.3. Analysis on Property Directed Reachability Algorithm. Property directed reachability (PDR or IC3) is probably the most significant breakthrough in SAT based model checking. The aim of PDR is to show that the system is *safe* w.r.t. the property θ; that is, each reachable state satisfies the boolean constraint θ.

The process of PDR retains a sequence $\Gamma_0,\ldots,\Gamma_k$ of clause sets fulfilling the following.

(1) $\Gamma_0=\{\theta_0\}$; and for each $i\leqslant k$ we have

(2) $\vdash\wedge\Gamma_i\ \to\ \wedge\Gamma_{i+1}$;

(3) $\vdash\wedge\Gamma_i\wedge\rho\ \to\ \wedge\Gamma'_{i+1}$;

(4) $\vdash\wedge\Gamma_i\ \to\ \theta$.

This process stops and reports an affirmative answer if there exists some $i>0$ having $\Gamma_i=\Gamma_{i-1}$; and it reports failure when some reachable "unsafe" state is detected. Therefore, it intends to find a (intermediate) reachable-closure enclosed (or *inductive set*) with θ, as described in Algorithm 2.

This algorithm employees two important subroutines—the strengthen process (Line 7) and the propagate process (Line 10):

(i) The process strengthen aims to disprove the reachability of the point $\mathscr{U}$ which violates θ. If it succeeds, some new constraints would be added to some specific clause sets; if this process fails, then it implies that we have a feasible path which leads to an "unsafe" state. In [21], an effective strengthen algorithm is provided, and it results in a linear complexity in time w.r.t. the size of variable set.

(ii) The process propagate is used to "compact" the clause sets and to accelerate the termination. Its purpose is to "push out" the clauses in a set as possible as they can. Suppose that there is a clause $\eta\in\Gamma_i$, in the case that $\vdash\wedge\Gamma_i\wedge\rho\ \to\ \eta'$ holds; since we have $\vdash\wedge\Gamma_i\wedge\rho\ \to\ \wedge\Gamma'_{i+1}$, it implies that $\vdash\wedge\Gamma_i\wedge\rho\ \to\ \wedge\Gamma'_{i+1}\wedge\eta'$ also holds. In such situation, the clause η could be definitely propagated to Γ_{i+1}.

Since the PDR algorithm could not directly handle general temporal properties, we need to first translate the LTL model checking problem to PDR solving, via the following steps.

(1) Build the tableau $T_{\neg\varphi}$ from the LTL formula φ, as described in Section 4.1.

(2) Thus, the problem is boiled down to a fair-circle detection problem on the production of M and $T_{\neg\varphi}$.

(3) This problem could be further translated to $n+1$ PRD-solving problems, where n is the number of fairness constraints in the production. Interested readers may refer to [22] for the translation details.

It could be seen that the features affecting the overhead of PDR-solving boiled from LTL input are the following:

(1) the scale of product system, which is determined by the number of variables in the tableau and model,

```
Input: The components $\mathscr{V}$, $\rho$, $\theta_0$ of the model $M$; a safety property $\theta$.
Output: The affirmative answer if $M$ is safe w.r.t. $\theta$; otherwise, a counterexample witnessing that $\neg\theta$ is reachable.
(1) let $k := 0$;
(2) let $\Gamma_0 = \{\theta_0\}$;
(3) let $\mathbf{Q} := \emptyset$; /* Q is a priority queue */
(4) repeat
(5)     while there exists $\mathscr{U} \subseteq \mathscr{V}$ s.t. $\mathscr{U} \Vdash \Gamma_k \wedge \neg\theta$ do
(6)         add $(\mathscr{U}, k)$ to the head of Q;
(7)         if strengthen($\{\Gamma_i\}_{i\leqslant k}, \mathbf{Q}, \mathscr{V}$) fails then
(8)             return counterexample extracted from Q;
(9)         end
(10)        propagate($\{\Gamma_i\}_{i\leqslant k}$);
(11)        if there exists some $j \leq k$ s.t. $\Gamma_j = \Gamma_{(j-1)}$ then
(12)            return "$M$ is safe from $\theta$";
(13)        end
(14)        let $\Gamma_{k+1} := \emptyset$;
(15)        $k := k + 1$;
(16)    end
(17) until $1 \neq 1$;
```

Algorithm 2: Framework of the PDR algorithm.

(2) the size (or number of clauses) of the transition relation, which is a summation of these of the tableau and the model,

(3) the number of fairness constrains in the product, which is also the summation of constraint numbers of the model and the tableau.

For two LTL formulae φ and ψ such that $\varphi \approx_M \psi$, since the model is the same, the "variable number," "number of clauses in the transition relation," and "the number of fairness contraints" are determined by the tableaux $T_{\neg\varphi}$ and $T_{\neg\psi}$. Precisely like the analysis we have done in Section 4.1, we have the same conclusions shown in Propositions 2 and 3 and Theorem 4, and these would explain why we can also benefit from CePRe when using PDR as underlying checking approach.

5. Experimental Results

We have implemented CePRe as an upfront option in NuSMV (the tool is available on http://sourceforge.net/projects/nusmvwithcepre/), and we have also conducted experiments upon both industrial benchmarks and randomly generated cases in terms of BDD-based model checking, bounded model checking, and PDR-based model checking.

The BMC encoding approach we adopt here is that proposed in [15], which is the current BMC implementation of NuSMV. Since PDR verifiers could not directly take SMV models and/or LTL formulae as input, and they use AIGs (the And-Inverter Graphs) as standard input format, we first use the SMVtoAIG tool [23] to convert SMV-scripts into AIGs and then use iimc [24] to perform the PDR-based verification.

We conduct the experiments under such platform: CPU-Intel Core Duo2 E4500 2.2 GHz, Mem-2 G Bytes, OS-Ubuntu 10.04 Linux, Cudd-v2.4.1.1, Zchaff-v2007.3.12.

5.1. Experiments upon Industrial Benchmarks. The benchmarks we choose in this paper are suggested in [4], and most of them come from real hardware verification.

Table 9 provides the experimental results for BDD-based LTL symbolic model checking. The field time is the executing time totally elapsed, and the field R.S. refers to the number of reachable states. For the experiments "with CePRe," both the overheads of time and space are the summations of preprocessing and model checking. For Table 9, we have the following remarks.

(1) 8 out of 16 specifications could be reduced with CePRe (and these specifications are marked with †).

(2) For the specifications that can be reduced, considerable improvements are made during verification. For example, for the specification Pit.g.ltl, with CePRe, the number of BDD nodes is decreased to 12.5% of that without using CePRe.

(3) When a specification cannot be reduced with CePRe, it spends a very low extra overhead for doing preprocessings.

(4) Something noteworthy we do not provide here is that in the case that a violated LTL specification can be reduced, the newly generated counterexample is usually shorter than that of before. Among 8 specifications that can be reduced, counterexample lengths of Pti.nuv.ltl, Pit.g.ltl, P0.ltl and Seq.ltl are, respectively, shortened to 15, 10, and 194, opposing to the original values 16, 12, and 217. Meanwhile, counterexample lengths of others are kept unchanged.

Table 10 gives the experimental results for BMC-based model checking, and we here give some comments on that.

(1) With NuSMV, we need to preset a max-bound when doing bounded model checking. The column max-bound gives such values—a "star mark" means that

Table 9: Comparative results of BDD-based MC with/without cepre.

Model	Spec.	Without cepre			With cepre		
		BDD--Nodes	R.S.	Time (sec.)	BDD--Nodes	R.S.	Time (sec.)
srg5	Ptimo.ltl[†]	7946	720	0.024	2751	720	0.016
	Ptignv.ltl[†]	29704	11460	0.058	5712	2880	0.012
	Pti.g.ltl[†]	64749	130048	0.048	8119	32768	0.016
abp4	P2false.ltl	99577	559104	0.200	99625	559104	0.202
	P2true.ltl[†]	61209	904384	0.066	56494	419296	0.064
	Pold.ltl	52301	353536	0.060	52349	353536	0.064
	Ptimo.ltl	78098	219616	0.080	78146	219616	0.088
	Pti.g.ltl	8385	200704	0.060	8433	200704	0.062
dme3	P0.ltl[†]	889773	35964	5.756	527983	26316	5.096
	P1.ltl	455148	8775	0.460	409432	5505	0.374
dme5	Mdl.ltl[†]	793942	$8.64316e+06$	167.346	814494	$3.2097e+06$	114.599
	Wat.ltl[†]	412867	$1.79217e+07$	302.005	967033	$1.12567e+07$	286.850
	Ptimo.neg	508036	$1.26202e+06$	3.260	508081	$1.26202e+06$	3.280
msi_w-trans	Sched.ltl	2275558	$7.31055e+07$	6.612	2275655	$7.31055e+07$	6.632
	Safety.ltl	1213308	$3.6528e+07$	7.568	1213460	$3.6528e+07$	7.644
	Seq.ltl [†]	1921973	$3.5946e+07$	93.570	1702585	$1.7973e+07$	94.085

Table 10: Experimental results of BMC-based MC with/without cepre.

Model	Spec.	Without cepre		With cepre		Max-bound
		N.O.C.	Time (sec.)	N.O.C.	Time (sec.)	
srg5	Ptimo.ltl[†]	272567	67.391	1371	0.143	20
	Pti.gnv.ltl[†]	2101	0.116	299	0.024	6
	Pti.g.ltl[†]	21	0.016	21	0.016	1
abp4	P2false.ltl	7532	3.972	7532	3.972	17
	P2true.ltl[†]	12639	8.145	9369	7.753	20[*]
	Pold.ltl	7499	9.087	7499	9.488	20[*]
	Ptimo.ltl	6332	2.500	6332	2.512	16
	Pti.g.ltl	11952	0.841	11952	0.976	20[*]
dme3	P0.ltl[†]	—	—	35102	524.207	62
	P1.ltl	216	0.036	167	0.048	1
dme5	Mdl.ltl[†]	90	0.044	90	0.048	0
	Wat.ltl[†]	367	0.048	274	0.052	1
	Ptimo.neg	367	0.050	277	0.058	1
msi_w-trans	Sched.ltl	14235	1.076	14235	1.078	20[*]
	Safety.ltl	12439	8.441	12439	8.448	20[*]
	Seq.ltl[†]	1907	0.064	81	0.052	3

this bound does not reach the completeness threshold. The field N.O.C. designates the number of clauses generated during model checking.

(2) From Table 10, we can see that without CePRe the specification Pti.gnv.ltl generates 2101 clauses when the verification stops; in contrast, it only produces 299 clauses if CePRe is switched on.

(3) Another comparison is for P0.ltl upon dme3: If we do not do any reduction, the SAT solver reports a SEGMENTATION FAULT at Step 35. In contrast, using CePRe, a counterexample could be found at Step 62.

(4) Since the encoding approach we use is taken from [15], propositions used in the encoding are only determined by the model and the bound; thus

Table 11: Experimental result of PDR-based MC with/without cepre.

Model	Spec.	PDR time without cepre (sec.)	PDR time with cepre (sec.)	Extratime for cepre (sec.)
srg5	Ptimo.ltl†	0.117	0.056	<0.001
	Pti.gnv.ltl†	0.084	0.043	<0.001
	Pti.g.ltl †	0.070	0.032	<0.001
abp4	P2false.ltl	3.187	3.185	0.004
	P2true.ltl†	2.996	1.239	0.004
	Pold.ltl	6.937	6.939	<0.001
	Ptimo.ltl	0.910	0.912	<0.001
	Pti.ltl	5.812	5.816	<0.001
dme3	P0.ltl†	9.815	6.086	<0.001
	P1.ltl	0.147	0.142	0.004
dme5	Mdl.ltl†	3.072	1.731	0.004
	Wat.ltl†	4.575	3.043	0.004
	Ptimo.neg	0.073	0.073	0.004
msi_wtrans	Sched.ltl	0.135	0.136	<0.001
	Safety.ltl	0.200	0.201	0.004
	Seq.ltl†	0.091	0.045	0.006

the number of required propositions does not change. For this reason, the corresponding experimental results on proposition numbers are not provided.

Experimental results of PDR-based model checking are shown in Table 11.

(1) For PDR-based model checking, we are mainly concerned about the time-overhead. Since that the PDR algorithm consumes memory evenly during verification, we do not provide the space-overhead here.

(2) Because the CePRe process is done in an individual phase in the PDR-based verification, we here also list the overhead for doing CePRe.

(3) We can see that a significant performance improvement could be made if we use CePRe in PDR-based model checking, and the extra overhead for preprocessing is still relatively negligible.

Note that both model-independent and model-dependent rules contribute to the reductions. For example, for the model srg5 and the specification Pti.g.ltl, the rules (FS) and (S) are applied; meanwhile, for the model msi_wtrans and the specification Seq.ltl, the application of $(\mathrm{U}^{\mathrm{U}}[\neg 2 \rightarrow 3])$ is invoked.

5.2. Experiments w.r.t. Random Models and Specifications. We have also performed experiments upon randomly generated models and specifications with the tool Lbtt [25] and with the methodology suggested in [17].

For BDD-based MC and bounded model checking, we conduct the comparison in the following manner. For each $3 \leq \ell \leq 7$, we randomly generate 40 specifications having length ℓ. Subsequently, for each specification, we generate two models, respectively, for the BDD-based model checking and for BMC. Hence, we totally have 200 specifications and 400 models.

For the BDD-based model checking, we give the comparative results on (1) the scale of BDD-nodes, (2) the number of reachable states, and (3) the time consumed, and the experimental results are, respectively, shown in Figures 1, 2, and 3. For bounded model checking, we have set the maximum bound to 20 and we have compared (1) the number of clauses and (2) the executing time; the results are, respectively, shown in Figures 4 and 5. Each value here we provide is the average of the 40 executions.

For the BDD-based model checking, there are 123 (out of 200) specifications that can be reduced, whereas for bounded model checking, the number of specifications that can be reduced is 118. Note that in this experiment, when CePRe is switched on, extra overheads (such as time) have also been taken into account.

For the PDR-based model checking, the experiments are conducted as follows. We first generate 50 SMV models with Lbtt and then we randomly generate LTL specifications for each model and each designated length. Next, we batchly do CePRe upon one group of specification copies and then obtain the product models (for each model and each specification). Subsequently, we convert the product models into AIG format with SMVtoAIG.

We here compare the number of verfication obligations that could be accomplished within the preset time bound (i.e., 600 sec.), and the results are shown in Figure 6. From that, we can see that with the increment of the specification's length, the ratio (that could be done with CePRe to without CePRe) also monotonically increases.

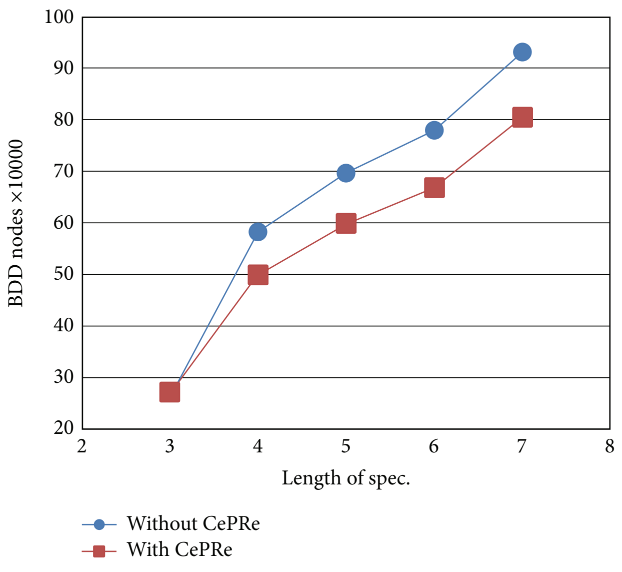


Figure 1: Results on the scale of BDD nodes in random BDD-MC experiments.

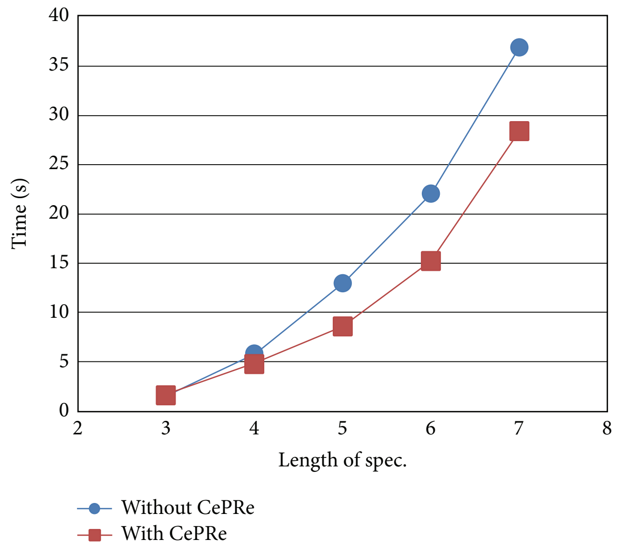


Figure 3: Time overhead in random BDD-MC experiments.

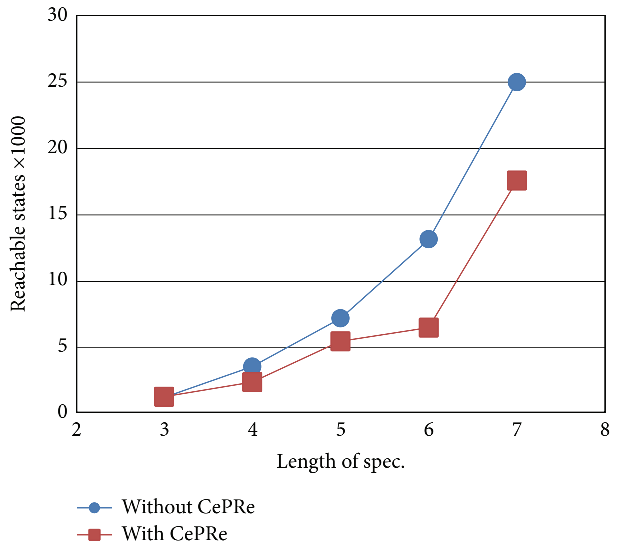


Figure 2: Results on reachable states in random BDD-MC experiments.

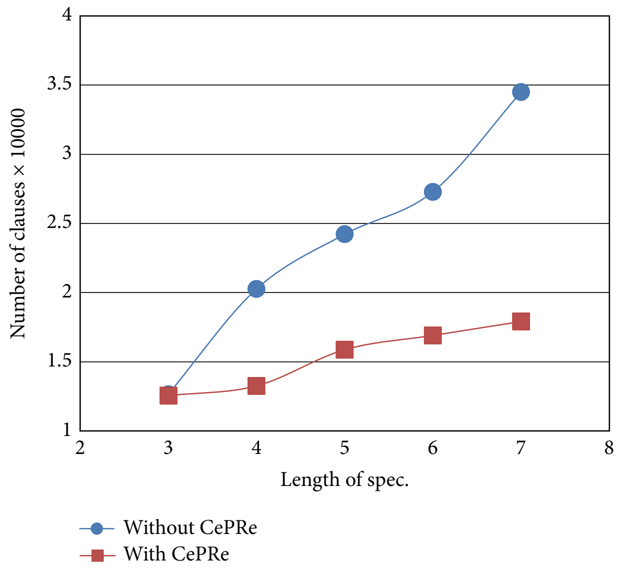


Figure 4: The scale of clauses in random BMC experiments.

6. Concluding Remarks

In this paper, we present a new technique to reduce LTL specifications' complexity towards symbolic model checking, namely, CePRe. The novelty in this technique is that the reduced formula needs not be logically equivalent with the original one but just preserves the counterexample set. Moreover, the condition enabling such a reduction can be usually detected with lightweight approaches, such as SAT-solving. Hence, this technique could leverage the power of SAT-solvers.

The central part of CePRe is a set of reduction rules, and soundness of these reduction rules is fairly easy to check. For the model-dependent rules, additional conditions mainly concern the invariants and transitions, and we do not make a sufficient use of other features, such as fairness. In this paper, the rules are given by enumerating all possible combinations of (at most two) temporal operators. Indeed, there might be some other reduction schemas we are not aware of.

From the experimental results, we can see that in a statistical perspective, a better performance and lower overhead can be achieved with CePRe. For the future work, we would consider how to extend our idea to symbolic model checking upon rich assertional languages, such as PSL.

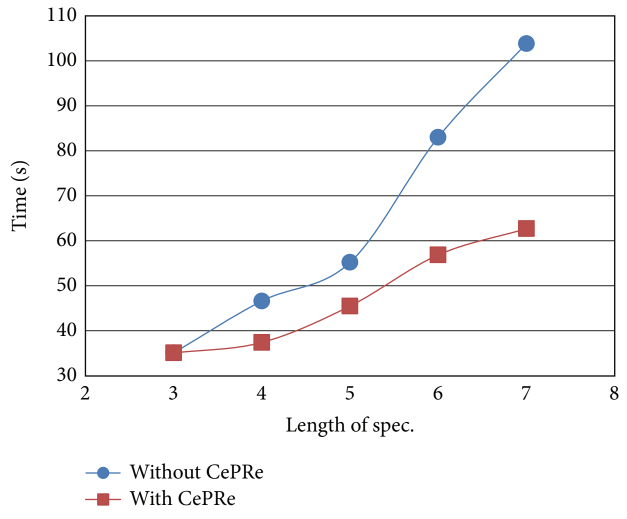


Figure 5: Time overhead in random BMC experiments.

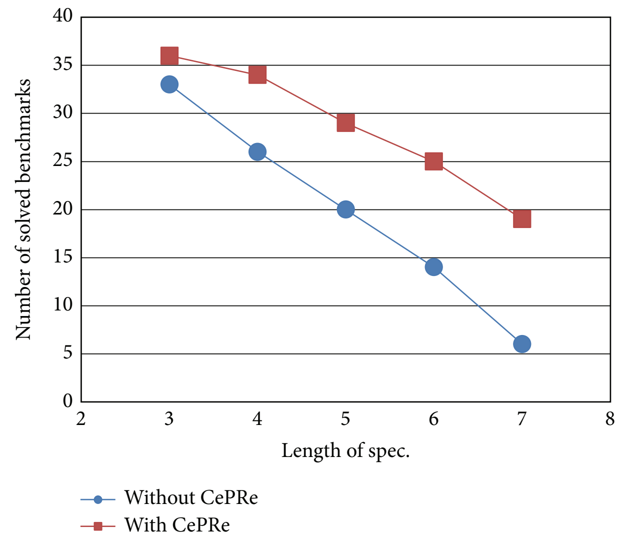


Figure 6: Experimental results of random PDR-based model checking.

Conflict of Interests

A conference version of this paper [26] is firstly published in the proceeding of ICTAC 2013, and a corresponding unpublished version is put on CoRR (http://arxiv.org/abs/1301.3299). The authors have extended more than 1/3 new material in this submission. The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

This paper is the extension of [26], and the authors would to thank the anonymous reviewers for the helpful comments of their conference version. This work is supported by NSFC (No. 61103012), the 973 Project (no. 2014CB340703), NSFC (nos. 61379054 and 61120106006), and the Program for New Century Excellent Talents in University.

References

[1] A. Pnueli, “The temporal logic of programs,” in *Proceedings of the 18th Annual Symposium on Foundations of Computer Science (FOCS ’77)*, pp. 46–57, IEEE Computer Society, 1977.

[2] M. Y. Vardi, “Branching vs. linear time: final showdown,” in *Tools and Algorithms for the Construction and Analysis of Systems*, vol. 2031 of *Lecture Notes in Computer Science*, pp. 1–22, Springer, 2001.

[3] F. Somenzi and R. Bloem, “Efficient B üchi automata from LTL formulae,” in *Computer Aided Verification*, E. A. Emerson and A. P. Sistla, Eds., vol. 1855 of *Lecture Notes in Computer Science*, pp. 53–65, Springer, 2000.

[4] A. Biere, K. Heljanko, T. Junttila, T. Latvala, and V. Schuppan, “Linear encodings of bounded LTL model checking,” *Logical Methods in Computer Science*, vol. 2, no. 5, pp. 1–64, 2006.

[5] K. L. McMillan, *Symbolic model checking, an approach to the state explosion problem [Ph.D. thesis]*, Carnegie Mellon University, Kluwer Academic Publishers, 1993.

[6] E. M. Clarke, O. Grumberg, and K. Hamaguchi, “Another look at LTL model checking,” in *Formal Methods in System Design*, vol. 818 of *Lecture Notes in Computer Science*, pp. 415–427, Springer, 1994.

[7] A. Biere, A. Cimatti, E. M. Clarke, and Y. Zhu, “Symbolic model checking without BDDs,” in *Proceedings of the 5th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS ’99)*, vol. 1579 of *Lecture Notes in Computer Science*, pp. 193–207, Springer, 1999.

[8] A. R. Bradley, “SAT-based model checking without unrolling,” in *Verification, Model Checking, and Abstract Interpretation*, vol. 6538 of *Lecture Notes in Computer Science*, pp. 70–87, Springer, 2011.

[9] F. Somenzi and A. R. Bradly, “IC3: where monolithic and incremental meet,” in *Proceedings of the International Conference on Formal Methods in Computer-Aided Design (FMCAD ’11)*, P. Bjesse and A. Sloodova, Eds., pp. 3–8, FMCAD, 2011.

[10] N. Een, A. Mishchenko, and R. Brayton, “Efficient implementation of property directed reachability,” in *Proceedings of the Formal Methods in Computer-Aided Design (FMCAD ’11)*, pp. 125–134, Austin, Tex, USA, November 2011.

[11] A. R. Bradley, “Understanding IC$_3$,” in *Theory and Applications of Satisfiability Testing—SAT 2012*, vol. 7317 of *Lecture Notes in Computer Science*, pp. 1–14, Springer, 2012.

[12] J. R. Burch, E. M. Clarke, K. L. McMillan, D. L. Dill, and L. J. Hwang, “Symbolic model checking: 10^{20} states and beyond,” *Information and Computation*, vol. 98, no. 2, pp. 142–170, 1992.

[13] D. Gabbay, “The declarative past and imperative future. Executable temporal logic for interactive systems,” in *Temporal Logic in Specification*, vol. 398 of *Lecture Notes in Computer Science*, pp. 409–448, Springer, 1989.

[14] O. Kupferman and M. Y. Vardi, “Modular model checking,” in *Compositionality: The Significant Difference*, vol. 1536 of *Lecture Notes in Computer Science*, pp. 381–401, Springer, 1998.

[15] A. Cimatti, M. Pistore, M. Roveri, and R. Sebastiani, “Improving the encoding of LTL model checking into SAT,” in *Verification, Model Checking, and Abstract Interpretation*, vol. 2294 of *Lecture Notes in Computer Science*, pp. 196–207, Springer.

[16] T. Latvala, A. Biere, K. Heljanko, and T. Junttila, “Simple bounded LTL model checking,” in *Formal Methods in Computer-Aided Design*, A. Hu and A. Martin, Eds., vol. 3312 of *Lecture Notes in Computer Science*, pp. 186–200, Springer, 2004.

[17] T. Latvala, A. Biere, K. Heljanko, and T. Junttila, “Simple is better: efficient bounded model checking for past LTL,” in *Verification, Model Checking, and Abstract Interpretation*, vol. 3385 of *Lecture Notes in Computer Science*, pp. 380–395, Springer, 2005.

[18] A. Zbrzezny, “A new translation from ETCL* to SAT,” in *Proceedings of the International Workshop CS&P*, M. Szczuka, Ed., pp. 589–600, September 2011.

[19] E. Clarke, D. Kroening, J. Ouaknine, and O. Strichman, “Completeness and complexity of bounded model checking,” in *Verification, Model Checking, and Abstract Interpretation*, vol. 2937 of *Lecture Notes in Computer Science*, pp. 85–96, Springer, 2004.

[20] A. Frisch, D. Sheridan, and T. Walsh, “A fixpoint encoding for bounded model checking,” in *Formal Methods in Computer-Aided Design*, vol. 2517 of *Lecture Notes in Computer Science*, pp. 238–255, 2002.

[21] A. R. Bradley and Z. Manna, “Checking safety by inductive generalization of counterexamples to induction,” in *Proceedings of the Formal Methods in Computer Aided Design (FMCAD ’07)*, pp. 173–180, November 2007.

[22] A. R. Bradley, F. Somenzi, Z. Hassan, and Y. Zhang, “An incremental approach to model checking progress properties,” in *Proceedings of the Formal Methods in Computer-Aided Design (FMCAD ’11)*, pp. 144–153, November 2011.

[23] AIGER, the SMVtoAIG toolkit, 2007, http://fmv.jku.at/aiger/.

[24] The IIMC tool, 2013, http://ecee.colorado.edu/wpmu/iimc/.

[25] H. Taurainen and K. Heljanko, “Testing LTL formula translation into Büchi automata,” *International Journal on Software Tools For Technology Transfer*, vol. 4, no. 1, pp. 57–70, 2002.

[26] W. Liu, R. Wang, X. Fu et al., “Conterexamplepreserving reduction for symbolic model checking,” in *Proceedings of the 10th International Colloquium on Theoretical Aspects of Computing (ICTAC ’13)*, Z. Liu, J. Woodcock, and H. Zhu, Eds., vol. 8049 of *Lecture Notes in Computer Science*, pp. 249–266, Springer, Shanghai, China, 2013.

Hindawi Publishing Corporation
Journal of Applied Mathematics
Volume 2014, Article ID 850789, 5 pages
http://dx.doi.org/10.1155/2014/850789

Research Article

Approximate Equivalence of the Hybrid Automata with Taylor Theory

Anping He,[1,2] Jinzhao Wu,[2] Shihan Yang,[2] and Hongyan Tan[3]

[1] *Chengdu Institute of Computer Application, Chinese Academy of Sciences, Chengdu 610041, China*
[2] *Guangxi Key Laboratory of Hybrid Computation and IC Design Analysis, Guangxi University for Nationalities, Guangxi 530006, China*
[3] *Institute of Acoustics, Chinese Academy of Sciences, Beijing 100190, China*

Correspondence should be addressed to Jinzhao Wu; hidrwu@sohu.com

Received 26 January 2014; Revised 7 April 2014; Accepted 22 April 2014; Published 11 May 2014

Academic Editor: Guiming Luo

Hybrid automaton is a formal model for precisely describing a hybrid system in which the computational processes interact with the physical ones. The reachability analysis of the polynomial hybrid automaton is decidable, which makes the *Taylor* approximation of a hybrid automaton applicable and valuable. In this paper, we studied the simulation relation among the hybrid automaton and its Taylor approximation, as well as the approximate equivalence relation. We also proved that the Taylor approximation simulates its original hybrid automaton, and similar hybrid automata could be compared quantitatively, for example, the approximate equivalence we proposed in the paper.

1. Introduction

Hybrid automaton is a formal model of a hybrid system in which computational processes interact with the physical ones. Similar to other automata, the hybrid one also contains states and transitions, but it labels and groups all closely related states as an activity to express continuous behaviour by a *state function*; moreover, it also employs functions to update variable values while transiting from an activity to another. In short, a hybrid automaton shows the hybrid behaviour composed of the discrete state transitions and continuous state evolution.

Most hybrid systems are safety critical applications, which require guarantees of safe operations; the unwarrantable configurations must be unreached. Formally, we can validate these safety properties by *reachability* analysis, for example, the forward or backward iteration with initial conditions, activities, and so on. However, for most types of hybrid automata, the computation of the iteration may not halt and then leads to an undecidable reachability analysis. Luckily, there exists one type of hybrid automaton called the polynomial hybrid automaton; its reachability analysis is proved decidable. Then it is very valuable to construct an approximate polynomial hybrid automaton of a common one. In [1], the authors build the polynomial one in terms of Taylor theory.

In this paper, we study the relation of the hybrid automaton and its Taylor approximation in terms of the semantic models, for example, how different similar automata behave, and then the quantitative comparison of the automata. We find that the automaton and its approximation show a simulation relation; furthermore, fortunately, we propose a quantitative equivalence of similar hybrid automata in terms of simulation. To our knowledge, there seems no work done for this.

1.1. Related Works. The hybrid automaton was introduced in [2, 3] with analysis for the linear and nonlinear one and some simple examples, and, in [4], the authors focus on its verification aspect. There are many works on verification and analysis of hybrid automata; [2, 4–6] studied the model checking of hybrid automata; [2, 3, 7] performed reachability analysis; [8, 9] showed some achievements of probabilistic hybrid automata; [10] studied the hybrid automata by a domain-theoretic semantics.

The behavior of hybrid system is so complex that the approximation of the system is necessary in most of applications. In [2], authors found the reachability of the multirated simple hybrid automata is decidable. Several papers (see, e.g., [1, 2, 7, 11, 12]) tried to classify hybrid automata, for example, a specific type of automaton satisfies the properties if its approximation holds the same ones. Other papers (see, e.g., [13–15]) were concerned with the way of analyzing classes of hybrid systems for which the strategy previously described does not work.

In [1], the authors approximated syntactically an automaton H with $\text{Approx}(H, k)$ in terms of Taylor theory and then studied also how close the behaviors of H and $\text{Approx}(H, k)$ are. However, they missed the relation between H and $\text{Approx}(H, k)$ in a behaviour view, nevertheless, the quantitative equivalence based on the relation was not considered either.

1.2. Organization of the Paper. The paper is organized as follows: in Section 2 we recall the definition of hybrid automata and their Taylor approximations; in Section 3 we study the simulation and approximation and then prove a hybrid automaton being simulated by its Taylor approximation; in Section 4 we propose a Taylor k-relation and then prove this relation is an equivalence; we concluded in Section 5.

2. Taylor Approximation of Hybrid System

Let us show the formal definition of a hybrid automaton.

Definition 1. A hybrid automaton H is a tuple $(\vec{x}, \phi_{\text{init}}, Q, q_0, \Sigma, T, \text{Act}, \text{Event})$.

(i) $\vec{x} = (x_1, \ldots, x_n)$ is vector of real variables.

(ii) $\phi_{\text{init}} \in \Phi((x_1, \ldots, x_n))$ is initial condition.

(iii) Q is finite set of states.

(iv) $q_0 \in Q$ is the initial state.

(v) Σ is a set of events symbol.

(vi) $\text{Act } Q \rightarrow \Phi((x_1, \ldots, t, x'_1, \ldots, x'_n))$ is the activity function assigning to each state q a formula $\text{Act}(q)$. The variable t represents time elapsing.

(vii) $T \subset Q \times \Phi((x_1, \ldots, x_n, t, x'_1, \ldots, x'_n)) \times Q$ is a set of transitions. Variables $x'_1, \ldots, x'_n$ represent the new values taken by the variables $x_1, \ldots, x_n$ after the transition: $T = T_f \cup T_j$:

  (a) $T_f \subset q \times \text{Act}(q) \times q$, that $q \in Q$,

  (b) $T_j \subset Q \times \Phi((x_1, \ldots, x_n, 0, x'_1, \ldots, x'_n)) \times Q$.

(viii) $\text{Event } T \rightarrow \Sigma$ is event mapping function attaching each transition an event name.

There are many similar definitions of hybrid automata, such as ones in [2, 3]. Generally, the hybrid automata behave in two ways, namely, an internal *flow* and state *jump* transition. The flow is a stutter at states which shows the continuous behavior of a hybrid automaton, for example, a variation that does not involve states switching but updates value of variables only. In contrast, jump transition always switches states. Let $q, q' \in Q$, $\alpha \in \Sigma$, and $t \in R$ be time elapsing. We denote the flow as below:

$$q \xrightarrow{t} q \tag{1}$$

and jump

$$q \xrightarrow{\alpha} q'. \tag{2}$$

The basic way of analyzing a hybrid automaton is *reachability analysis*, which involves computing the forward or backward reachable regions of a hybrid automaton from initial conditions recursively [1, 2]. However, in most cases, the computing failed because of the nondecidability of the reachabilty analysis.

There are many aspects that can be adopted to classify the hybrid automata; one of them concerns the mathematical expressions; for example, a hybrid automaton is a *polynomial hybrid automaton* if and only if ϕ_{init} is a polynomial formula; for each state q, $\text{Act}(q)$ is a polynomial formula, and for each transition (q, ϕ, q'), ϕ is a polynomial formula. Researchers and engineers try their best to find the polynomial hybrid automaton, or even an approximated one, of a real system, because the problem of reachability in n steps for polynomial hybrid automata is decidable [16, 17].

It is clear that the most types of hybrid automata do not belong to the polynomial ones; then it is valuable to study a polynomial approximation of a hybrid automaton. Ruggero Lanotte and Simone Tini proposed polynomial approximation of a generic hybrid automaton in [1] in terms of Taylor approximation theory. The following is the definition of Taylor approximation of a hybrid automaton H.

Definition 2. Let H be a hybrid automaton; all functions appearing in its formulae Φ are derivable $k+1$ times. The approximation of rank k for $H = (\vec{x}, \phi_{\text{init}}, Q, q_0, \Sigma, T, \text{Act}, \text{Event})$ is the polynomial hybrid automaton $\text{Approx}(H, k) = (\vec{x}, \phi_{\text{init}_k}, Q', q'_0, \Sigma, T_k, \text{Act}_k, \text{Event}')$.

(i) Each formula in ϕ_{init_k}, T_k, and Act_k is derived from ϕ by replacing each nonpolynomial subformula $f(\vec{y}) \sim c$ in ϕ with

$$\begin{aligned} &p^k(f, \vec{y}) - R^k(f, \vec{y}, \phi) \sim c, \quad \text{if } R^k(f, \vec{y}, \phi) \neq \infty; \\ &\qquad \text{true}, \quad \text{if } R^k(f, \vec{y}, \phi) = \infty, \end{aligned} \tag{3}$$

where $\vec{y}$ stays for either $\vec{x}$, if ϕ is ϕ_{init}, or $(\vec{x}, \vec{x}')$, if ϕ is a transition label, or $(\vec{x}, t, \vec{x}')$, if ϕ is an activity function.

(ii) Q' is constructed by one-to-one mapping M, which maps Q to Q', $q'_0 = M(q_0)$.

(iii) $T_k = \{(q_{1_k}, \phi_k, q_{2_k}) \mid q_{1_k} = M(q_1) \wedge q_{2_k} = M(q_2) \wedge (q_1, \phi, q_2) \in T\}$.

(iv) $\text{Act}_k = \{(q_k, \phi_k) \mid q_k = M(q) \wedge (q, \phi) \in \text{Act}\}$.

(v) $\text{Event}' = \{(t_k, \alpha)\}$ that $(t, \alpha) \in \text{Event}$ with $t \in T \wedge t_k \in T_k \wedge \alpha \in \Sigma$ and t corresponds to t_k.

In [1], the authors also proved that given any hybrid automaton H and $k \in \mathbb{N}$, the polynomial hybrid automaton $\text{Approx}(H, k)$ is an approximation of H.

We study the relations between generic hybrid automata and their Taylor approximations.

3. The Simulation of a Hybrid Automaton and Its Approximation

The *behavior* is one of the most important concepts in formal methods, which shows a way of analyzing a formal system dynamically. The behavior analysis is also seen as the base of comparing formal systems. Formally, the behavior is composed of traces of sequences of states, which exhibits the system run. We study the relation between hybrid automata by their behaviors.

There are two kinds of hybrid system run, namely, flow and jump, which interleave and interact. Starting from initial condition, the hybrid automaton shows state sequence, which we called *trace*. Similar to the discrete systems, the simplest comparison of hybrid automata is a trace equivalent, for example, whether hybrid automata own the same trace set. However, the trace equivalence is not an accurate way of comparison; it cannot distinguish different behaviors under the branching semantics. The hybrid system is composed of computational processes interacting with physical processes; it rejects the simple comparison which may hide the key information. In contrast, simulation relation checks each state in each trace starting from the initial; it is considered that simulation is more accurate for the complex branching semantics.

Definition 3. Let $H_1 = (\vec{x}, \phi_{\text{init}_1}, Q_1, q_{0_1}, \Sigma, T_1, \text{Act}_1, \text{Event}_1)$ and $H_2 = (\vec{x}, \phi_{\text{init}_2}, Q_2, q_{0_2}, \Sigma, T_2, \text{Act}_2, \text{Event}_2)$ be hybrid automata with the same event symbols. The binary relation $\succ \subseteq Q_1 \times Q_2$ is a simulation of H_2 by H_1 if the following two conditions hold.

(i) For each state $q_1 \in Q_1$, $q_2 \in Q_2$, and $t \in R$ being time elapsing, if $q_1 \preceq q_2$ and $q_1 \xrightarrow{t}_1 q_1$, then there exists a state q_2' such that $q_2 \xrightarrow{t}_2 q_2'$ and $q_1' \preceq q_2'$.

(ii) For each state $q_1 \in Q_1$, $q_2 \in Q_2$, and $\alpha \in \Sigma$, if $q_1 \preceq q_2$ and $q_1 \xrightarrow{\alpha}_1 q_1$, then there exists a state q_2' such that $q_2 \xrightarrow{\alpha}_2 q_2'$ and $q_1' \preceq q_2'$.

(iii) For initial state q_{0_2} and for state $q_1 \in Q_1$, if $q_1 \preceq q_{0_2}$, then $s_1 = q_{0_1}$.

We denote H_1 is simulated by H_2 as $H_1 \preceq H_2$ for short.

Definition 3 shows a qualitative relation that the hybrid automaton H_2 simulates H_1. Because the hybrid systems are so complex that it is not easy to be analyzed directly, it is a common way of computing the reachability of the approximation. Let us define the approximation of a hybrid automaton formally; the following definition is from [1].

Definition 4. A hybrid automaton H' is an approximation of a hybrid automaton H if H' is obtained from H by replacing each formula ϕ with a formula ϕ' such that $[\![\phi]\!] \subseteq [\![\phi']\!]$.

The operator $[\![\,]\!]$ maps a formula to its satisfied value set, and it is proved in [1] that the $\text{Approx}(H, k)$ is the approximation of a hybrid automaton H.

The Taylor approximation is one of the most useful and applicable approximation of a hybrid automaton. Although the Taylor approximation has been proposed for several years, the relation between the approximation and hybrid automaton is not well studied. It is interesting to check the simulation relation between an automaton and its Taylor approximation. We show this as a theorem.

Theorem 5. *A hybrid automaton is simulated by its Taylor approximation, for example, for a hybrid automaton H, $H \preceq \text{Approx}(H, k)$.*

Proof. Suppose $H = (\vec{x}, \phi_{\text{init}}, Q, q_0, \Sigma, T, \text{Act}, \text{Event})$ is a hybrid automaton and $\text{Approx}(H, k) = (\vec{x}, \phi_{\text{init}_k}, Q', q_0', \Sigma, T_k, \text{Act}_k, \text{Event}')$ its Taylor approximation. Let $q \in Q$ and $q' \in Q'$; q and q' have relation R, for example, qRq'; we prove R is a simulation relation.

Case 1 ($q \xrightarrow{\alpha} q_1$ that $\alpha \in \Sigma$ be an event). According to Definition 2, Q' is constructed by a one-to-one mapping from Q; for example, $\exists q_k, q_{1_k} \in Q'$ that $q_k = M(q)$, $q_{1_k} = M(q_1)$, and $q_k \xrightarrow{\alpha} q_{1_k}$.

Case 2 ($q \xrightarrow{t} q$ that $t \in R$ be time elapsing). According to Definition 2, $\exists q_k \in Q'$ that $q_k = M(q)$, $\text{Act}_k(q_k) = \phi_k$, and $\text{Act}(q) = \phi$. Then according to Definition 4, we have $[\![\phi]\!] \subseteq [\![\phi_k]\!]$. Let $\vec{u} = (u_1, \ldots, t_0, u_1', \ldots, u_n')$ and $\vec{u} = (u_1, \ldots, t_0 + t, u_1', \ldots, u_n')$ be two values that satisfy $\text{Act}(q)$ at moments t_0 and $t_0 + t$ separately; then t is a time duration; for example, we formally express this as $\vec{u} = (u_1, \ldots, t_0, u_1', \ldots, u_n')$, $\vec{u} = (u_1, \ldots, t_0 + t, u_1', \ldots, u_n') \in [\![\phi]\!]$. Because we have known that $[\![\phi]\!] \subseteq [\![\phi_k]\!]$, then $\vec{u} = (u_1, \ldots, t_0, u_1', \ldots, u_n')$, $\vec{u} = (u_1, \ldots, t_0 + t, u_1', \ldots, u_n') \in [\![\phi_k]\!]$; for example, we get $q_k \xrightarrow{t} q_k$.

Case 3 ($q_{0_k} = M(q_0)$ in terms of Definition 2). Then Theorem 5 is proved. □

From now on, we know that the Taylor approximation simulates its original hybrid automaton; for example, the behavior of the hybrid automaton is preserved by its Taylor approximation totally. Although there might be "granularity" problems while checking the safety related properties, the approximation provides a very applicable way. Moreover, Theorem 5 could be extended to all types of approximations.

4. Approximate Equivalence Relation

Theorem 5 shows a relation between simulation and approximation. Since approximation of a hybrid automaton is always quantifiable, we can study a quantified simulation relation indirectly.

Two hybrid automata could be almost the same although their formulae may be different; for example, events are same; formula owns the same Taylor expansion regardless of their remainder. We call this relation *Taylor k-related*; k is the rank of this relevancy that shows that all formulae are derivable $k+1$ times. We can construct an upper bound in terms of upper bounds of remainders of two Taylor related hybrid automata and then construct a hybrid automaton that simulates both two systems.

Theorem 6. *The Taylor k-related hybrid automata are simulated by the same hybrid automaton.*

Proof. Suppose $H_i = (\vec{x}, \phi_{\text{init}_i}, Q_i, q_{0_i}, \Sigma, T_i, \text{Act}_i, \text{Event}_i)$ with $i = \{1, 2\}$; H_1 and H_2 are Taylor k-related.

We can construct the Taylor approximation of each hybrid automaton, $\text{Approx}(H_i, k)$. According to Theorem 5, $H_i \preceq \text{Approx}(H_i, k)$. Let Φ_i be the formulae set of H_i and Φ_{i_k} the set of $\text{Approx}(H_i, k)$; then according to Definition 2, for each $\phi_i \in \Phi_i$ and $\phi_{i_k} \in \Phi_{i_k}$, we have

$$\phi_{i_k} = \begin{cases} P_i^k(f_i, y_i) - R_i^k(f_i, y_i, \phi_i) \sim c, \\ \quad \text{if } R_i^k(f_i, y_i, \phi_i) \neq \infty; \\ \text{true}, \\ \quad \text{if } R_i^k(f_i, y_i, \phi_i) = \infty. \end{cases} \quad (4)$$

Because H_1 and H_2 are Taylor k-related, then $P = P_1^k(f_1, y_1) = P_2^k(f_2, y_2)$ and $\text{Event}_1((q_1, \phi_1, q_1')) = \text{Event}_2((q_2, \phi_2, q_2'))$ with $\phi_i \in \Phi_i$.

Now we construct the hybrid automaton $H = (\vec{x}, \phi_{\text{init}}, Q, q_0, \Sigma, T, \text{Act}, \text{Event})$ with

$$\phi_i = \begin{cases} P^k(f, y) - R \sim c, & \text{if } R_i^k(f_i, y_i, \phi_i) \neq \infty; \\ \text{true}, & \text{if } R_i^k(f_i, y_i, \phi_i) = \infty. \end{cases} \quad (5)$$

R is an upper bound of $R_1^k(f_1, y_1, \phi_1)$ and $R_2^k(f_2, y_2, \phi_2)$ that $\forall \vec{u} \in [\![R_1^k(f_1, y_1, \phi_1)]\!] \cup [\![R_2^k(f_2, y_2, \phi_2)]\!]$ and $\vec{u} \in [\![R]\!]$, and $\text{Event}((q, \phi, q')) = \text{Event}((q_1, \phi_1, q_1'))$.

We prove H is an approximation of H_i; for example, prove that $[\![\phi_i]\!] \subseteq [\![\phi]\!]$ for any formula ϕ. We have assumed that $\sim$ is in $\{<, \leq\}$. Let us begin with the base case where $\phi_i = P_i^k(f_i, \vec{y}_i) - R_i^k(f_i, \vec{y}_i, \phi_i) \sim c$ and $\phi = P - R \sim c$.

Let $\vec{u}$ be a vector such that $\vec{u} \in [\![\phi_i]\!]$; that is, $f_i(\vec{u}) \sim c$. Let $r_i^k(f, \vec{u}, \vec{0})$ be Lagrange remainder of ϕ_i; for example,

$$r_i^k(f_i, \vec{u}, \vec{0}) = \sum_{j_1 + \cdots + j_n = k+1} \frac{\left(\left(D_1^{j_1} \cdots D_n^{j_n} f\right)(\vec{0})\right) \cdot u_1^{j_1} \cdots u_n^{j_n}}{j_1! \cdots j_n!}. \quad (6)$$

According to Definition 2, $r_i^k(f_i, \vec{u}, \vec{0}) \in [-R_i^k(f_i, \vec{u}, \phi_i), R_i^k(f_i, \vec{u}, \phi_i)]$. By previous analysis, there exists some remainder $r^k(f, \vec{u}, \vec{0}) \in [-R, R]$. Therefore,

$$\left(\vec{u}, r_i^k(f_i, \vec{u}, \vec{0})\right) \in [\![P + e \sim c \land e \in [-R, R]]\!] \quad (7)$$

which is equivalent to

$$\left(\vec{u}, r_i^k(f_i, \vec{u}, \vec{0})\right) \in [\![e \sim c - P \land e \in [-R, R]]\!]. \quad (8)$$

Since $e \geq -R$ and $\sim \in \{\leq, <\}$, the last equation above implies

$$\left(\vec{u}, r_i^k(f_i, \vec{u}, \vec{0})\right) \in [\![-R \sim c - P \land e \in [-R, R]]\!] \quad (9)$$

which is equivalent to

$$\left(\vec{u}, r_i^k(f_i, \vec{u}, \vec{0})\right) \in [\![P - R \sim c \land e \in [-R, R]]\!], \quad (10)$$

which implies that $\vec{u} \in [\![\phi]\!]$.

Then let us consider the cases of formulae composition, for example, formulae connected with $\land$ or/and $\lor$. Let us consider the inductive step $\phi_i \equiv \phi_{i_1} \lor \phi_{i_2}$. It holds that $\phi \equiv \phi_1 \lor \phi_2$. By the inductive hypothesis, $[\![\phi_{i_1}]\!] \subseteq [\![\phi_1]\!]$ and $[\![\phi_{i_2}]\!] \subseteq [\![\phi_2]\!]$. Hence, $[\![\phi_i]\!] = [\![\phi_{i_1}]\!] \cup [\![\phi_{i_2}]\!] \subseteq [\![\phi_1]\!] \cup [\![\phi_2]\!] = [\![\phi]\!]$. The case of $\land$ is similar and not shown here.

Then we proved that $[\![\phi_i]\!] \subseteq [\![\phi]\!]$; for example, H is an approximation of H_i. According to Theorem 5, H simulate H_i. Then Theorem 6 is proved. □

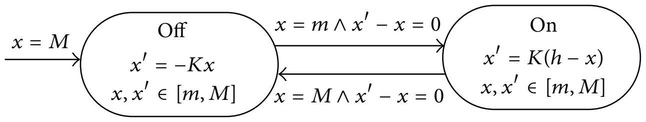


FIGURE 1: Hybrid automaton for a thermostat.

We call the Taylor approximation of Taylor k-related hybrid automata *Taylor related approximation*. It is easy to see that the Taylor related relation is an equivalence, for example, a relation satisfying reflexive, transitive and symmetry.

Theorem 7. *The Taylor related approximation is an approximate equivalence, which is denoted as $\asymp_k$ with k being the rank of relevancy.*

Then finally we propose a quantified approximate equivalence.

We can compare two similar hybrid systems by their Taylor approximation, for example, a quantitative comparison that k shows the degree of the similarity. Let us study a thermostat that is described in [2]; the temperature of a room is controlled by a thermostat which continuously senses the temperature and turns a heater on and off. When the heater is off, the temperature describes in terms of the function $x(t) = \theta e^{-Kt}$; when the heater is on, the temperature follows $x(t) = \theta e^{-Kt} + h(1 - e^{-Kt})$, where h and K are constants. We can express the thermostat formally in Figure 1.

According to Theorem 5, we can construct its approximation (see Figure 3 in [1]); the thermostat is simulated by its approximation; for example, let the hybrid automaton of thermostat be denoted as H and let its approximate be $\text{Approx}(H, 3)$; then $H \preceq \text{Approx}(H, 3)$. Furthermore, in terms of Theorem 7, it is easy to know that $H \asymp_3 \text{Approx}(H, 3)$.

5. Conclusion

In this paper, we studied the simulation relation among the hybrid automata and their Taylor approximations and then proposed an approximate equivalence relation. The simulation relation discovers how a Taylor approximation confirms to its original hybrid automaton; meanwhile, the equivalence explores the degree of similarity of similar automata quantitatively. In future, we plan studying the bisimulation of the automata and approximations in two ways, a more accurate approximation theory and metric semantics of hybrid automata.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

This work is partly supported by the NSF of Guangxi nos. 2011GXNSFA018154, 2012GXNSFGA060003, and 2013GXNSFAA019342, the Science and Technology Foundation of Guangxi no. 10169-1, Guangxi Scientific Research Project no. 201012MS274, GUN Project no. 2012Q017, and the Bagui scholarship project.

References

[1] R. Lanotte and S. Tini, "Taylor approximation for hybrid systems," *Information and Computation*, vol. 205, no. 11, pp. 1575–1607, 2007.

[2] R. Alur, C. Courcoubetis, N. Halbwachs et al., "The algorithmic analysis of hybrid systems," in *Theoretical Computer Science*, vol. 138, pp. 3–34, 1995.

[3] T. A. Henzinger, P.-H. Ho, and H. Wong-toi, "Algorithmic analysis of nonlinear hybrid systems," *IEEE Transactions on Automatic Control*, vol. 43, no. 4, pp. 540–554, 1998.

[4] T. A. Henzinger, "The theory of hybrid automata," Technical Report UCB/ERL M96/28, EECS Department, University of California, Berkeley, Calif, USA, 1996.

[5] R. Gentilini, K. Schneider, and B. Mishra, "Successive abstractions of hybrid automata for monotonic CTL model checking," in *Proceedings of the International Symposium on Logical Foundations of Computer Science (LFCS '07)*, pp. 224–240, June 2007.

[6] A. Podelski and S. Wagner, "Model checking of hybrid systems: from reachability towards stability," in *Hybrid Systems: Computation and Control*, vol. 3927 of *Lecture Notes in Computer Science*, pp. 507–521, Springer, Berlin, Germany, 2006.

[7] T. A. Henzinger, P. W. Kopke, A. Puri, and P. Varaiya, "What's decidable about hybrid automata?" *Journal of Computer and System Sciences*, vol. 57, no. 1, pp. 94–124, 1998.

[8] B. C. Williams and M. M. Henry, *Model-based estimation of probabilistic hybrid automata [M.S. thesis]*, Howard University, Washington, DC, USA, 2002.

[9] J. M. B. Braman and R. M. Murray, "Probabilistic safety analysis of sensordriven hybrid automata," *Hybrid Systems: Computation and Control*, 2009.

[10] A. Edalat and D. Pattinson, "Denotational semantics of hybrid automata," in *Proceedings of the Foundations of Software Science and Computation Structure (FoSSaCS '06)*, pp. 231–245, 2006.

[11] R. Alur, T. A. Henzinger, G. Lafferriere, and G. J. Pappas, "Discrete abstractions of hybrid systems," *Proceedings of the IEEE*, vol. 88, no. 7, pp. 971–984, 2000.

[12] T. A. Henzinger and R. Majumdar, "Symbolic model checking for rectangular hybrid systems," in *Proceedings of the Tools and Algorithms for the Construction and Analysis of Systems (TACAS '00)*, Lecture Notes in Computer Science, pp. 142–156, Springer, New York, NY, USA, 2000.

[13] A. M. Bayen, E. Cruck, R. M. Bayen, and C. J. Tomlin, "Guaranteed overapproximations of unsafe sets for continuous and hybrid systems: solving the hamilton-jacobi equation using viability techniques," in *Hybrid Systems: Computation and Control*, pp. 90–104, Springer, New York, NY, USA, 2002.

[14] A. Chutinan and B. H. Krogh, *Verification of Polyhedral-Invariant Hybrid Automata Using Polygonal Flow Pipe Approximations*, Springer,, New York, NY, USA, 1999.

[15] A. B. Kurzhanski and P. Varaiya, "On reachability under uncertainty," *SIAM Journal on Control and Optimization*, vol. 41, no. 1, pp. 181–216, 2002.

[16] M. Fränzle, "Analysis of hybrid systems: an ounce of realism can save an infinity of states," in *Computer Science Logic*, vol. 1683 of *Lecture Notes in Computer Science*, pp. 126–140, Springer, Berlin, Germany, 1999.

[17] M. Fränzle, "What will be eventually true of polynomial hybrid automata?" in *Theoretical Aspects of Computer Software*, vol. 2215 of *Lecture Notes in Computer Science*, pp. 340–359, Springer, Berlin, Germany, 2001.

Hindawi Publishing Corporation
Journal of Applied Mathematics
Volume 2014, Article ID 369350, 9 pages
http://dx.doi.org/10.1155/2014/369350

Research Article

Multitask Oriented Virtual Resource Integration and Optimal Scheduling in Cloud Manufacturing

Zhen Cheng,[1] Dechen Zhan,[1] Xibin Zhao,[2] and Hai Wan[2]

[1] *School of Computer Science and Technology, Harbin Institute of Technology, 92 West Dazhi Street, Nangang, Harbin 150001, China*
[2] *School of Software, Tsinghua University, NLIST, KLISS, Beijing 100084, China*

Correspondence should be addressed to Zhen Cheng; chengzhen_hit@163.com

Received 13 February 2014; Accepted 2 April 2014; Published 4 May 2014

Academic Editor: Xiaoyu Song

To deal with the problem of resource integration and optimal scheduling in cloud manufacturing, based on the analyzation of the existing literatures, multitask oriented virtual resource integration and optimal scheduling problem is presented from the perspective of global optimization based on the consideration of sharing and correlation among virtual resources. The correlation models of virtual resources in a task and among tasks are established. According to the correlation model and characteristics of resource sharing, the formulation in which resource time-sharing scheduling strategy is employed is put forward, and then the formulation is simplified to solve the problem easily. The genetic algorithm based on the real number matrix encoding is proposed. And crossover and mutation operation rules are designed for the real number matrix. Meanwhile, the evaluation function with the punishment mechanism and the selection strategy with pressure factor are adopted so as to approach the optimal solution more quickly. The experimental results show that the proposed model and method are feasible and effective both in situation of enough resources and limited resources in case of a large number of tasks.

1. Introduction

Cloud manufacturing is a typical model of service-oriented manufacturing [1]; its core idea is to form the pool of cloud manufacturing resource based on the virtualization for manufacturing resources and servitization for manufacturing capacity through the integration of new technologies such as cloud computing and IoT and then provide various manufacturing services which can be obtained anytime, used on demand, safe and reliable, of high-quality, and of low-cost for the full life-cycle manufacturing process, and finally to achieve customers, manufacturers, and providers value added jointly [2, 3].

In cloud manufacturing service system, based on cloud manufacturing service platform, manufacturing enterprises can integrate a large number of personalized requirements to form clustering requirements and can form a dynamic entity named cloud enterprise [4] according to dynamic configuration of clustering requirements and integration of virtual manufacturing resources from the pool of cloud manufacturing resources. The cloud enterprise can provide products or services on demand and finally can satisfy personalized requirements through the optimal scheduling of virtual resources and the mapping of virtual resources and real resources.

After receiving customers' requirements, cloud manufacturing enterprise usually fulfills through several steps: (1) task classification; (2) task decomposition; (3) resource allocation; (4) resource optimal scheduling; (5) task execution and services feedback evaluation. A large number of existing literatures are based on single task [5–8]; however, tasks are received continuously in reality. If the method for single task problem is used to solve multitask problem, there would be some following shortages that the scheduling result makes every single task optimized but might not be the best in global view.

(1) If multiple tasks are scheduled one by one adopting the existing single task oriented scheduling method simply, we should sort tasks using typical strategy (e.g., first-come-first-service, task priority marked, etc.) firstly and then configure

optimal schedule resources for each task in turn. This makes the optimization goal that the configuration and optimal scheduling of resources only meet the current task each time. The problem is that if the key resources can meet lots of tasks, but only used in the priority task, this will lead to the optimization for multiple tasks from a global perspective that cannot be achieved. Even though the literature [9] optimized the problem from the global view, it did not consider the relationship of resources which would be emphasized in this paper.

(2) Only the exclusivity of resources is considered, and the sharing of resources is not taken into account in a scheduling process. The exclusivity of resources means that if a resource is occupied or used, it cannot be used by other tasks at the same time. In the related literatures of service computing and/or cloud computing [9, 10], the exclusivity of resources means that a resource is used only once in a scheduling process, no matter how many tasks are in a scheduling process. And also the exclusivity of resources is a typical characteristic of resources scheduling in service computing and/or cloud computing environment, because the unit of time for a scheduling is usually seconds or even milliseconds, microseconds [11, 12]. However, in cloud manufacturing environment, the unit of time for resources scheduling is usually days or weeks, so in order to enhance the resources utilization, resources should be considered to reuse in other tasks when resources are free in a multiple tasks scheduling process. In this paper, we call the time division sharing of resources.

(3) The optimal scheduling did not consider the correlation of resources. Resource scheduling, service selection, discovery, and composition in the literatures are based on the features of resources and/or services [13–15]. However, in fact, there are correlations among resources, particularly resources were used in tasks. Normally, for example, two consecutive subtasks in a task performed by resources of the same provider should be more efficient than different providers. Here, the same provider is the correlation of resources.

In cloud manufacturing environment, tasks may be continuous, but, in this paper, we put time into segments and handle all tasks in a time slot just like planning in manufacturing enterprises. And then we could optimally schedule virtual resources at the beginning of a time slot, fully considering the correlation and sharing of resources, to break through the limitation of local optimization and to achieve the overall optimization for multiple tasks.

2. Problem Formulation

Just like service composition and optimal scheduling for single task, every single task of multiple tasks for optimal scheduling in cloud manufacturing also should go through decomposition, production of candidate resource sets, selection of resources, and selection of execution path. From the perspective of the execution path of subtasks and candidate resource sets, tasks will be divided into 3 categories in this paper.

(1) Different types of tasks: the execution path of subtasks and corresponding candidate resource sets are totally different. For example, tasks from logistics enterprises and tasks from manufacturing enterprises are different types.

(2) Same type of tasks: the execution path of subtasks and corresponding candidate resource sets are totally the same. For example, multiple tasks from a certain type of product manufacturing enterprises are the same type.

(3) Mixed types of tasks: the execution path of subtasks and corresponding candidate resource sets are partly the same. For example, multiple tasks from different types of product manufacturing enterprises are mixed types. All the tasks have different manufacturing processes but require the same cutting process.

Virtual resources integration and optimal scheduling for different types of tasks could adopt the method of traditional single task execution because of totally different candidate resource sets and execution paths. And for mixed types of tasks, we will study it in the future because of its complexity. So in this paper, we will focus on the same type of tasks and study virtual resource integration and optimal scheduling.

2.1. Motivating Example. Before introducing the problem, an example of manufacturing process of C70E open wagon in CNR is presented to illustrate the problems that are addressed in this paper. The task of manufacturing process for C70E open wagon, denoted by $Ta(\mathit{deliverTime, price})$, has five subtasks such as wheel processing, hook lock processing, body processing, assembling, painting, experiment, and delivery. Assuming the candidate set of virtual resources (VRS) for each process has three candidate virtual resources (VR), as shown in Figure 1, the first number of the candidate resource denotes the processing days and the second number denotes the cost. For example, the first number "5" of virtual resource $VR_1^1(5, 700)$ indicates the processing days and the second number "700" indicates the cost.

When a cloud manufacturing enterprise received 3 tasks: $\{Ta_1(30, 5000), Ta_2(25, 8000), Ta_3(15, 10000)\}$, we can get an optimal execution path of resource integration $\{P_1, P_2, P_3\}$ to maximize profit $\text{Max} \sum_{i=1}^{3} \text{Profit}(P_i)$, using traditional heuristic method without considering the sharing and correlation of resources. However, when the cloud manufacturing enterprise receives the fourth task $Ta_4(20, 9000)$ at the same time, the tasks could not be fully scheduled without considering the sharing of resources. The general method is that we could select 3 tasks $\{Ta_2, Ta_3, Ta_4\}$ from the 4 tasks to maximize profit. But if we consider the sharing of resources, the cloud manufacturing enterprise could fully schedule the 4 tasks

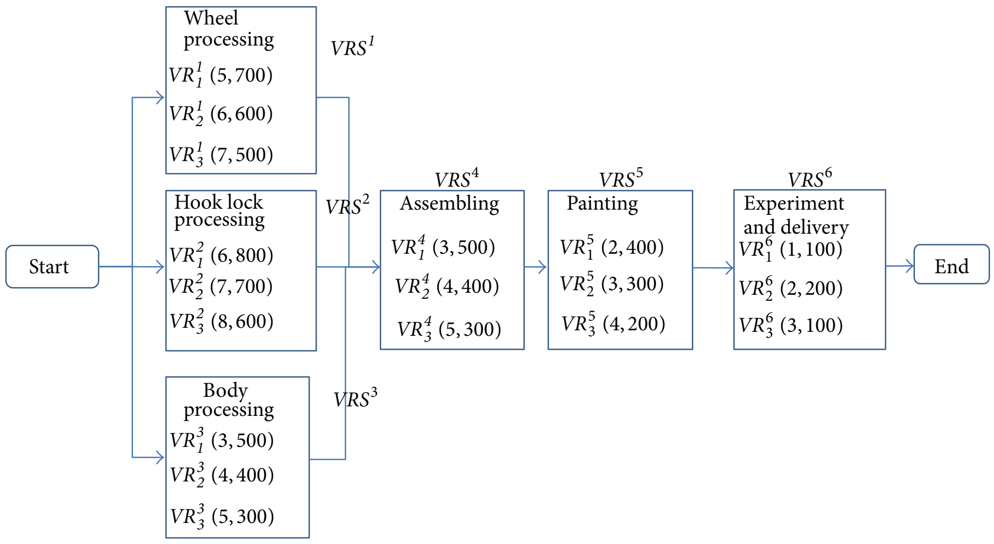


Figure 1: Manufacturing process of C70E open wagon.

before the deadline of each task. The following is a feasible solution (maybe not the best solution):

$$\begin{aligned} &P_1 \left\langle \left\langle VR_3^1, VR_3^2, VR_3^3 \right\rangle, VR_3^4, VR_3^5, VR_3^6 \right\rangle \\ &P_2 \left\langle \left\langle VR_2^1, VR_2^2, VR_2^3 \right\rangle, VR_2^4, VR_2^5, VR_2^6 \right\rangle \\ &P_3 \left\langle \left\langle VR_1^1, VR_1^2, VR_1^3 \right\rangle, VR_1^4, VR_1^5, VR_1^6 \right\rangle \\ &P_4 \left\langle t_{\text{wait}}^4, \left\langle VR_1^1, VR_1^2, VR_1^3 \right\rangle, VR_1^4, VR_1^5, VR_1^6 \right\rangle, \end{aligned} \quad (1)$$

where $t_{\text{wait}}^4 = \text{MAX}(VR_1^1, VR_1^2, VR_1^3) = 6$. The performance period of tasks is shown in Figure 2.

The feasible solution is that we just delayed starting the fourth task Ta_4 6 days ($t_{\text{wait}}^4 = 6$). The feasible solution is typically considering the time division sharing of resources ($VR_1^1, VR_1^2, VR_1^3, VR_1^4, VR_1^5, VR_1^6$). Obviously, the feasible solution could be further optimized.

On the other hand, there is correlation among resources from the manufacturing and service perspective. In this paper, we divided the correlation of resources into two categories.

(1) The correlation of resources in a task: we take the first task Ta_1, for example, if the selected resources of the last three subtasks belong to the same provider ($VR_3^4 \cdot$ provider $= VR_3^5 \cdot$ provider $= VR_3^6 \cdot$ provider), the cost of VR_3^4, VR_3^5, VR_3^6 will be lower than the default cost. If all of them have 10% off, the cost of execution path P_1 will decrease from 2000 to 1940.

(2) The correlation of resources among tasks: we take the feasible solution, for example, the resources $VR_1^1, VR_1^2, VR_1^3, VR_1^4, VR_1^5, VR_1^6$ are used twice, and then the cost will be reduced per use. Assume that all of them have 5% off and the cost of path P_3 will be reduced 150, the same as P_4. Finally, the cost of solution will be reduced by 300 than without correlation among tasks.

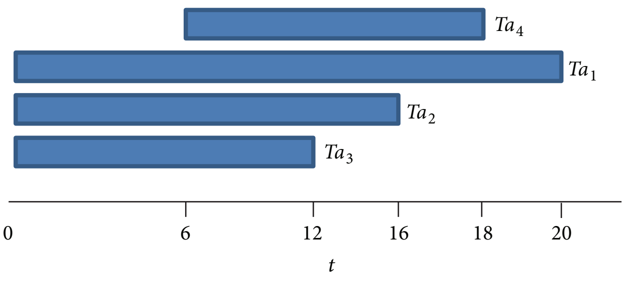


Figure 2: Multitask performance period.

So, the sharing and correlation of resources are fully considered to realize virtual resource integration and optimal scheduling for multitask in this paper.

2.2. Problem Formulation

2.2.1. Resource Correlation Modelling. For the correlation in a task, we use ontology to describe virtual resources firstly. The virtual resource model can be defined as a 3-tuple: $VR(TRAS, CRAS, OAS)$, where $TRAS$ denotes time related attributes of virtual resources, for example, capacity and resource provider. $CRAS$ denotes cost related attributes of virtual resources, for example, resource provider and the location of corresponding resources. OAS denotes the other attributes, for example, the size of corresponding resources. In this paper, we realize virtual resources integration and optimal scheduling for multitask from two aspects: time and cost. For the other attributes such as the reliability of virtual resources, we could also handle using the proposed method.

Secondly, we can model the correlation of resources based on the virtual resource model. We take the cost of virtual resources $C(VR)$ as an example to model the correlation of resources in a task, and the same as the time of virtual resources $T(VR)$:

$$C(VR) = \{\text{default} : \text{Value}_0; g_1(\overline{VRS}.\text{selected}.CRA) : \text{Value}_1; g_2(\overline{VRS}.\text{selected}.CRA) : \text{Value}_2; \ldots\}, \quad (2)$$

where Value_0 denotes the default value without considering the correlation and $\overline{VRS}$ denotes the other candidate virtual resource sets in which the virtual resource VR is not included. CRA denotes the cost related attributes; the Boolean function g is the correlation function of resources. Therefore, $g_1(\overline{VRS}\cdot\text{selected}\cdot CRA) : \text{Value}_1$ denotes that if the CRA of the selected resource in the $\overline{VRS}$ makes the Boolean function g_1 true, the value of the $C(VR)$ will be Value_1. Obviously, the Boolean function g has recursive operations; we only use NOT and AND operations as follows:

$$g_i() = \overline{g_j()}, \quad i \neq j;$$
$$g_i() = g_j() \&\& g_k(), \quad i \neq j \neq k. \quad (3)$$

For the OR and XOR operations, we can use NOT and AND operations to realize. And NOT operation is prior to AND operation. If there are multiple correlation functions to meet at the same time, the value of the $C(VR)$ will be selected as the minimized value because the smaller the cost is the better. For other cases, such as reliability, it is the bigger the better.

Therefore, in this paper, the calculation model for time and cost based on resource correlation in a task is as follows:

$$T(VR) = \text{Min}\{\text{Value}_0, \text{Value}_i\}, \quad \text{where } g_i() = \text{true}. \quad (4)$$

$$C(VR) = \text{Min}\{\text{Value}_0, \text{Value}_i\}, \quad \text{where } g_i() = \text{true}; \quad (5)$$

We will take a detailed example to illustrate the correlation of resources in a task. We still take the cost of VR_3^4 in the feasible solution P_1 as an example to model the correlation. Firstly, we build the ontology description model of VR_3^4 as follows: $VR_3^4(\{\cdots\}, \{CNR, QiQiHaer, \ldots\}, \{\cdots\})$, where the first set is $TRAS$, and the third set is OAS. The second set is the value of $CRAS$; that is to say the provider of VR_3^4 is CNR, and the location is $QiQiHaer$. Then, we can build the cost calculation model $C(VR_3^4)$ as follows:

$$C(VR_3^4) = \{\text{default} : 300; VRS^5.\text{selected.provider} = CNR : 285; VRS^5.\text{selected.provider} = CNR \;\&\&\; VRS^6.\text{selected.provider} = CNR : 270; VRS^5.\text{selected.location} = QiQiHaer : 295 \ldots\}. \quad (6)$$

In the above model, the default cost of using VR_3^4 is 300. If the provider of the selected resources of VRS^5 corresponding to the fifth subtask is also CNR, the cost of using VR_3^4 will be 285. And further, if the provider of the sixth subtask is also CNR, the cost of using VR_3^4 will be 270. If the location of the selected resources of VRS^5 corresponding to the fifth subtask is $QiQiHaer$, the cost of using VR_3^4 will be 295. Obviously, if the provider of the fifth subtask is the same as the sixth subtask and also is the CNR, the second and the third correlation functions in the above model are met at the same time. According to the features of cost, the minimized value 270 (the third correlation function) will be selected.

For the correlation among tasks, we also take the cost of virtual resources $C(VR)$ as an example to model the correlation of resources among tasks, and the same as the time of virtual resources $T(VR)$:

$$C(VR) = \{\text{default} : \text{defaultValue}; \text{correlationFunc}_1(\text{DefaultValue}) : \text{Value}_1; \text{correlationFunc}_2(\text{DefaultValue}) : \text{Value}_2; \ldots\}, \quad (7)$$

where DefaultValue is calculated by formula (5). $\text{correlationFunc}_1(\text{DefaultValue}) : \text{Value}_1$ denotes that if the selected resources in a scheduling process satisfy the correlation function correlationFunc_1, the value of the $C(VR)$ will be Value_1. For example, the cost per use may be reduced according to the increased times of use. And then, we can simply define the correlationFunc_1 as follows:

$$\text{correlationFunc}_1 = \text{DefaultValue} * \text{Dec}(VR)^{\text{Num}(VR)-1}, \quad (8)$$

where $\text{Num}(VR)$ is the times of using VR and $\text{Dec}(VR)$ denotes cost decay rate.

Again, in this paper, the calculation model for time and cost based on resource correlation among tasks is as follows:

$$T(VR) = \text{Min}\{\text{Value}_0, \text{Value}_i\},$$
$$C(VR) = \text{Min}\{\text{Value}_0, \text{Value}_i\}. \quad (9)$$

2.2.2. Problem Formulation.

2.2.2. Problem Formulation. Taking account of all the above features, we give the problem description as follows.

Let $Ta = \{Ta_1, Ta_2, \ldots, Ta_n\}$ denote the same type of tasks which the cloud manufacturing enterprise receives within a period of time. Each task has corresponding bid price $\{\text{Price}_1, \text{Price}_2, \ldots, \text{Price}_n\}$. Let $\{T_1, T_2, \ldots, T_n\}$ denote the number of days to delivery date for each task. And each task can be decomposed to J subtasks. Let $P = \{P_1, P_2, \ldots, P_m\}$ be a feasible solution. Each subtask can be executed by a resource of the corresponding candidate resource set VRS_j^i. And there are J_K resources $\{{}^1VR_j^i, {}^2VR_j^i, \ldots, {}^KVR_j^i\}$ in each candidate resource set VRS_j^i. Then the feasible solution should be under the following constraints:

$$m \leq n \tag{10}$$

$$T(P_i) \leq T(Ta(P_i)) \tag{11}$$

$$\text{if } {}^K VR_j^i [t_1, t_2] \in P_i, \qquad {}^K VR_j^i [t_1, t_2] \notin \bigcup P_p, \qquad i \neq p; \quad 1 \leq p \leq m. \tag{12}$$

The constraint (10) means that some tasks may be not executed. In constraint (11), $Ta(P_i)$ denotes the task corresponding to the feasible solution P_i. And constraint (11) means that all the executed tasks must be finished before delivery date. Constraints (12) mean that the exclusivity and sharing are considered for each selected resource. If the ${}^K VR_j^i$ was used by P_i in the period of $[t_1, t_2]$, it cannot be used in any other P_p $(i \neq p)$ at the same time, but it can be used in any other period.

How to evaluate the feasible solution? From the perspective of cloud manufacturing enterprise, the following objects should be considered.

(i) The more the profit the better:

$$\text{MAX} \sum_{i=1}^{m} \text{Profit}(P_i), \tag{13}$$

where $\text{Profit}(P_i) = \text{Price}(Ta(P_i)) - C(P_i)$.

(ii) The bigger the number of tasks that can be executed the better:

$$\text{MAX}(m). \tag{14}$$

The above two objects are not necessarily linear relations. For an extreme example, the profit of a task may be more than all the other tasks in an enterprise. In reality, it is also meaningful. For example, a start-up cloud manufacturing enterprise may reduce some profits to get more customers, then get more market share, and finally get more profits.

2.3. Model Analysis. Obviously, the problem of multitask oriented virtual resource integration and optimal scheduling in cloud manufacturing belongs to the multiobjective optimization problem. Such problems can usually be solved by Pareto optimal method and can also be transformed to single objective to solve by weighting method. In this paper, we transform the multiple objectives to single objective as follows by linear weighting method:

$$\text{MAX}\left(\alpha * \sum_{i=1}^{m} \text{Profit}(P_i) + \beta * m\right), \tag{15}$$

where α and β are the corresponding weighted parameters to control the relative significance of each objective, and $\alpha + \beta = 1$. The values will be given according to the target of the optimization.

In order to reduce the number of variables in the model and obtain more convenience to solve the problem, we normalize and handle the $\text{Profit}(P_i)$ as follows:

$$\overline{\text{Profit}(P_i)} = \frac{\text{Profit}(P_i) - \text{Min}(\text{Profit}(P_i))}{\text{Max}(\text{Profit}(P_i)) - \text{Min}(\text{Profit}(P_i))}. \tag{16}$$

The variables $X = (x_1, x_2, \ldots, x_n)$ are given, where x_i is Boolean value. If the value of x_i is 1, the task Ta_i will be executed. Otherwise, the task Ta_i will not be executed. And then (15) can be transformed as follows:

$$\text{MAX}\left(\alpha * \sum_{i=1}^{n} \left(\overline{\text{Profit}(Ta_i)} * x_i\right) + \beta * \sum_{i=1}^{n} x_i\right). \tag{17}$$

Therefore, (11) can be transformed as follows:

$$T(P_i) * x_i \leq T_i * x_i. \tag{18}$$

In order to simplify (12), it is assumed that we will schedule tasks in the future period T and divide T into several equal parts. Therefore, in any period $t \in T$, we will give the Boolean variable $y(i, j, k, t) \cdot y(i, j, k, t)$ which denotes that if the kth virtual resource in the candidate resource set VRS_j^i is selected to execute the task Ta_i in the period t, the variable $y(i, j, k, t)$ will be true. Otherwise, the variable $y(i, j, k, t)$ will be false. And then (12) will be transformed as follows:

$$\sum_{i=1}^{n} y(i, j, k, t) \leq 1, \quad \forall t \in T, j, k \text{ fixed}. \tag{19}$$

Meanwhile, the variable $y(i, j, k, t)$ should satisfy the following constraint to ensure that a subtask is only executed by a resource in the period t:

$$\sum_{k=1}^{J_k} y(i, j, k, t) \leq 1, \quad \forall t \in T, i, j \text{ fixed}. \tag{20}$$

Therefore, the mathematical formulations for multitask oriented virtual resource integration and optimal scheduling in cloud manufacturing are as follows: (17) subject to (18), (19), (20), and (21). Consider that

$$x_i \in \{0, 1\}, \quad 1 \leq i \leq n. \tag{21}$$

Apparently, the above problem belongs to NP-hard problem.

3. Virtual Resource Integration and Optimal Scheduling

GA is a population-based stochastic optimization technique inspired by the mechanism of natural selection and evolution. According to the code of solution space and the operation of the decision variables for the problem, GA can get the proximate optimal solution quickly by guided search in the total solution space. Therefore, GA is usually used to solve the problem of composition and optimal scheduling under the complex constraints.

In this paper, we adopt GA based on the real number matrix code to solve the problem of multitask oriented virtual resource integration and optimal scheduling in cloud manufacturing.

```
(1) For 1 to $T \times J$ // looped by columns
(2)   For 1 to $n$ //checked by rows
(3)     If $a_{ijt} = 0$, jump to Step 2;
(4)     Else
          If $VC_i = 0$, jump to Step 2;
          Else
            If not existed the same non-zero number, jump to
            Step 2;
            Else
              Randomly selected a number from the difference
              set between the $[0, J_k]$ set and the current column
              value set to replay. If the difference sets is Ø, then
              set $a_{ijt} = 0$;
(5)   For End
(6) For End
```

Algorithm 1

3.1. Chromosomal Representation. The solution is coded as the following real number matrix:

$$A = \begin{bmatrix} a_{111} & a_{121} & \cdots & a_{1J1} \cdots a_{11T} & a_{12T} & \cdots & a_{1JT} \\ a_{211} & a_{221} & \cdots & a_{2J1} \cdots a_{21T} & a_{22T} & \cdots & a_{2JT} \\ \cdots & \cdots & \cdots & \cdots\cdots\cdots & \cdots & \cdots & \cdots \\ a_{n11} & a_{n21} & \cdots & a_{nJ1} \cdots a_{n1T} & a_{n2T} & \cdots & a_{nJT} \end{bmatrix}. \quad (22)$$

Each row of matrix A denotes a task. The columns in matrix A are divided into T parts, each part indicates a period t, and, in each part, there are J columns which denote J subtasks in each task. Therefore, there are $T \times J$ columns in the matrix A, and the subscript meanings of the element a_{ijt} in the matrix A are as follows:

i: the identification of the ith task;

j: the jth subtask of the task;

t: the tth period.

The value of the matrix element a_{ijt} is an integer from $[0, J_k]$. It means that the a_{ijt}th virtual resource in the jth candidate resource set is selected to execute the task Ta_i in the period t where $a_{ijt} \neq 0$. Otherwise, the jth subtask of the task Ta_i in the period t will not be executed.

The matrix is related to the variables as follows.

(i) If the values of the ith row in the matrix are all 0, the ith task Ta_i will not be executed, and then the variable x_i equals 0. Otherwise, the value of the variable x_i is 1.

(ii) If $a_{ijt} \neq 0$ in the period t, the variable $y(i, j, a_{ijt}, t)$ equals 1, and then the values of all the other rows in the jth column of the tth part cannot equal a_{ijt} which is expressed as $y(i', j, a_{ijt}, t) = 0$, where $i' \neq i$. Obviously, the constraint (19) is satisfied.

(iii) If $a_{ijt} \neq 0$ in the period t, the variable $y(i, j, a_{ijt}, t)$ equals 1, and then for the same i and j, $y(i, j, k, t) = 0$, where $k \neq a_{ijt}$ and $k \in [0, J_k]$. Obviously, the constraint (20) is satisfied.

3.2. Initial Population. Each individual in the initial population is generated randomly column by column as follows so as to ensure the individual randomness and avoid the local optimization.

(i) In the period of $t = 0$, we used the following method to generate the first column value. If the number of tasks is less than the number of virtual resources in the corresponding candidate resources set, that is, $n \leq J_k$, we will randomly select n values from $[0, J_k]$ to form a random sequence as the first column. Otherwise, we will select all of the J_k values and $n - J_k$ zeros to form a random sequence as the first column. After that the value of all the other $J - 1$ columns in the period of $t = 0$ will be zero according to the rule that any task can be transformed to the sequence process [13].

(ii) In the period of $t > 0$, we firstly see the corresponding column of the period $t - 1$. If the jth subtask is not finished in the period $t - 1$, then $a_{ijt} = a_{ij(t-1)}$. If the subtask is finished, then $a_{ijt} = 0$ and $a_{i(j+1)t} \neq 0$ at the same time. The value of the $a_{i(j+1)t}$ should be randomly picked from the remaining numbers after being finished by $a_{i(j+1)t} = a_{i(j+1)(t-1)}$.

3.3. Crossover and Mutation. Because of the real number encoding of the solution, the traditional crossover and mutation operators of the Boolean value cannot be simply used in this paper. We design the crossover and mutation operators as follows.

(1) Row-Based Crossover. We will firstly generate an n-dimension Boolean type column vector VC. We will swap the ith row of the two parent matrices if $VC_i = 1$, do nothing if $VC_i = 0$, and finally generate the new two matrices. The constraints may not be satisfied, that there may be the same nonzero number in the same column after row-based crossover operating. Therefore, we must run the following fixed program to make the new two matrices satisfy the constraints. See Algorithm 1.

(2) Column-Based Crossover. Similar to the row-based crossover, we will firstly generate a $T \times J$-dimension Boolean

type row vector VC. We will swap the jtth row of the two parent matrixes if $VC_{jt} = 1$, do nothing if $VC_{jt} = 0$, and finally generate the new two matrixes. Obviously, the new two matrixes also satisfy the constraints.

(3) Block-Based Crossover. Block-based crossover is operated for the unit of block corresponding to the period of t. Therefore, we will generate a T-dimension Boolean type row vector VC. We will swap the tth block of the two parent matrixes if $VC_t = 1$, do nothing if $VC_t = 0$, and finally generate the new two matrices. Obviously, similar to the column-based crossover, the new two matrices also satisfy the constraints.

(4) Row-Based Mutation. Firstly, we give the mutation rules. We will change the value as follows if the value is not zero and do not change the value if the value is zero:

$$a_{ijt} = J_k - a_{ijt} + 1. \tag{23}$$

For the row-based mutation, we will firstly generate a t-dimension Boolean type column vector VC. We will change all of nonzero number according to the above rule in the ith row of the parent matrix if $VC_i = 1$, do nothing if $VC_i = 0$, and finally generate the new matrix. The constraints may not be satisfied that there may be the same nonzero number in the same column after row-based mutation operating. Therefore, we must run the above fixed program to make the new matrix satisfy the constraints.

(5) Column-Based Mutation. Similar to the row-based mutation, We will firstly generate a $T \times J$-dimension Boolean type row vector VC. We will change all of nonzero number according to the above rule in the jtth row of the parent matrix if $VC_{jt} = 1$, do nothing if $VC_{jt} = 0$, and finally generate the new matrix. Obviously, the new matrices also satisfy the constraints.

(6) Block-Based Mutation. Similar to the block-based crossover, block-based mutation is also operated for the unit of block corresponding to the period of t. Therefore, we will generate a T-dimension Boolean type row vector VC. We will change all of nonzero numbers according to the above rule in the tth block of the parent matrix if $VC_t = 1$, do nothing if $VC_t = 0$, and finally generate the new matrix. The constraints may not be satisfied that there may be the same nonzero number in the same column after block-based mutation operating. Therefore, we must run the above fixed program to make the new matrix satisfy the constraints.

3.4. Fitness Function and Selection Strategy. Based on the objective function, the fitness function will be given in this paper as follows:

$$F = \left(\alpha * \sum_{i=1}^{n} \left(\overline{\text{Profit}\left(Ta_i\right)} * x_i\right) + \beta * \sum_{i=1}^{n} x_i\right). \tag{24}$$

So as to select the new generation population, the penalty function for Ta_i will be given as follows:

$$pu_i = \begin{cases} 1, & \text{satisfied (18)} \\ \mu * \left(1 - \dfrac{T_i * x_i - T\left(P_i\right) * x_i}{\text{Max}\left(T_i\right)}\right), & \text{notsatisfied (18)}, \end{cases} \tag{25}$$

where μ denotes penalty factor. If the constraint (18) is satisfied in the task Ta_i, the penalty will not be carried out; that is to say the value of pu_i is 1. Otherwise, the penalty will be carried out, and if $x_i = 1$, the value of the $|T_i - T(P_i)|$ is bigger, and the penalty factor μ is also bigger. Therefore, we can give the penalty function for the feasible solution as follows:

$$Pu = \prod_{i=1}^{n} pu_i. \tag{26}$$

Based on the above function, the final fitness function in this paper is given as follows:

$$FF = F * Pu. \tag{27}$$

Apparently, we should reduce the number of the executed tasks or adjust the execution time or sequence of tasks if the constraint (18) is not satisfied in the feasible solution.

Traditional direct proportion selected strategy is adopted in this paper to generate offspring. In addition, we give the pressure factor for the fitness function so as to approach the optimal solution quickly.

For the individual s in the population whose size is *Size*, the fitness value is FF_s, and the selection probability is

$$\rho_s = \frac{FF_s - \chi * \text{Min}\,(FF)}{\sum_{s=1}^{Size} \left(FF_s - \chi * \text{Min}\,(FF)\right)}, \tag{28}$$

where χ denotes pressure factor, and the value range is $[0, 1)$.

4. Experiments and Discussion

These experiments are performed based on the above example (described in Section 2.1) to test the effectiveness of the proposed method. We will perform the experiments with the conditions of enough resources and limited resources.

4.1. Experiments and Discussion with Enough Resources. We will evaluate the results affected by the correlation of resources with enough resources. The experiments are performed in the following 3 situations:

(i) the correlations are not taken into account E1;

(ii) only the correlations in a task are taken into account E2;

(iii) both the correlations in a task and among tasks are taken into account E3.

The parameters of the model are given as follows: the population size $Size = 100$, max generation is 100, the probability of crossover is 0.9, the probability of mutation is 0.05, the pressure factor $\chi = 0.9$, the penalty factor $\mu = 0.01$, and $\alpha = 0.5$ and $\beta = 0.5$. And the percent of correlation resources in a task is 30%; the percent of correlation resources among tasks is 40%.

The results of the experiments are shown in Figure 3.

In Figure 3, the y-coordinate denotes the total profits of all executed tasks. The x-coordinate denotes the number

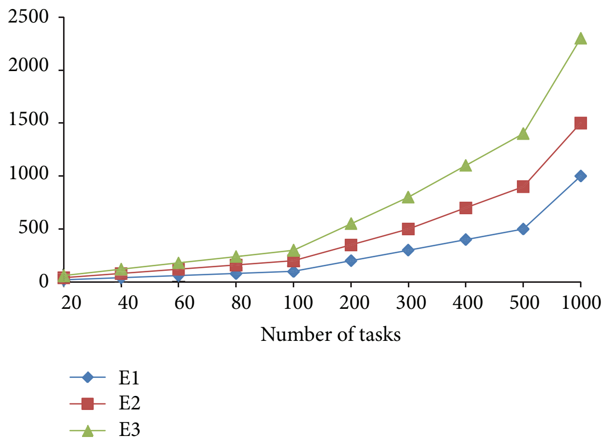


Figure 3: Results of the experiments with enough resources.

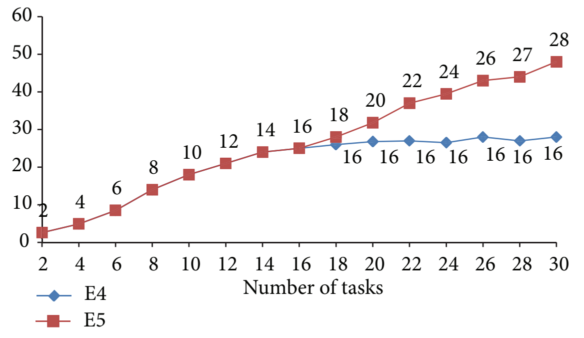


Figure 4: Results of the experiments with limited resources.

of tasks. With the number of tasks increased, the number of resources in corresponding candidate resources set is also increased to ensure that there are enough resources to execute all the tasks, but the percent of the correlation in a task and among tasks will not be changed.

As we can see in Figure 3, the total profit discrepancy in 3 situations is not bigger if the number of tasks is small (the number is less than 100). The reason is that the influence of the correlation of resources used in tasks is very small. But along with the increased number of tasks, the advantages of E2 and E3 which take correlations into account are huge, and the profits of E2 and E3 are much more than E1 for the same number of tasks, and the gap will be bigger and bigger. That is because the correlations both in a task and among tasks are considered, and the cost of selected resource will be lower than the default value. Therefore, it can be concluded that the proposed method that considered the correlations of resources has better performance than without considering the correlations.

4.2. Experiments and Discussion with Limited Resources. We will evaluate the results affected by the sharing of resources with limited resources. The experiments are performed in the following 2 situations:

(i) the sharing is not taken into account E4;

(ii) the sharing is taken into account E5.

The parameters of the model are given as follows: the population size $Size = 200$, max generation is 200, the probability of crossover is 0.95, the probability of mutation is 0.03, the pressure factor $\chi = 0.99$, the penalty factor $\mu = 0.01$, and $\alpha = 0.5$ and $\beta = 0.5$. And the percent of correlation resources in a task is 30%; the percent of correlation resources among tasks is 40%. The number of resources in each candidate resources set is fixed to 16, and the number of the given tasks varies from 2 to 30 which is an increment of 2.

The results of the experiments are shown in Figure 4.

In Figure 4, the y-coordinate denotes the total profits of the executed tasks. The x-coordinate denotes the number of given tasks. The number beside lines denotes the number of the executed tasks.

As we can see in Figure 4, the total profit and the number of executed tasks in E4 are the same as E5 if the number of tasks is less than 16. The reason is that all of the tasks could be executed without considering the sharing of resources. But along with the increased number of tasks, the number of tasks will not increase and keep 16 without considering the sharing of resources. But all the tasks can be executed with considering the sharing of resources until the number of given tasks is 28, and the profits are much more than without considering the sharing of resources. That is because the sharing of resources is considered. Therefore, it can be concluded that the proposed method that considered the sharing of resources has better performance than without considering the sharing and can solve the problem better for multitask oriented virtual resource integration and optimal scheduling which resources are limited.

5. Conclusions and Future Works

In this paper, the correlation and sharing of resources are both considered during multitask oriented virtual resources integration and optimal scheduling in cloud manufacturing, and the correlation and sharing models are presented. The problem is proposed and formulized, and GA based on the real number matrix code is employed to solve the problem. The experimental results show that the proposed methods are more effective and efficient for the problem.

Only the cost and time of Qos are considered in this paper; actually, there are many other factors to be considered such as reliability and trust. In the algorithm, there are a lot of invalid solutions in the solution space. In the future, we should avoid the invalid solutions during the process of algorithm or reduce the solution space by optimizing the code of solution to make the algorithm more effective. In addition, only one cloud manufacturing enterprise is considered; in the future, we should solve the problem for multienterprises.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

This paper is supported by the National Science and Technology Support Program (Grant no. 2012BAF12B16), the 973 Program of China (Grant no. 2010CB328000), the National Natural Science Foundation of China (Grant nos. U1201251 and 61133016), and the National 863 Plan of China (Grant no. 2012AA040906).

References

[1] Z. de-Chen, C. Zhen, Z. Xi-Bin et al., “Manufacturing service and its maturity model,” *Computer Integrated Manufacturing Systems*, vol. 18, no. 7, pp. 1584–1594, 2012.

[2] B.-H. Li, L. Zhang, S.-L. Wang et al., “Cloud manufacturing: a new service-oriented networked manufacturing model,” *Computer Integrated Manufacturing Systems*, vol. 16, no. 1, pp. 1–7, 16, 2010.

[3] X. Xu, “From cloud computing to cloud manufacturing,” *Robotics and Computer-Integrated Manufacturing*, vol. 28, no. 1, pp. 75–86, 2012.

[4] D.-C. Zhan, X.-B. Zhao, S.-Q. Wang et al., “Cloud manufacturing service platform for group enterprises oriented to manufacturing and management,” *Computer Integrated Manufacturing Systems*, vol. 17, no. 3, pp. 487–494, 2011.

[5] C. S. Hu, C. D. Xu, X. B. Cao, and J. Fu, “Study of classification and modeling of virtual resources in Cloud Manufacturing,” *Applied Mechanics and Materials*, vol. 121, pp. 2274–2280, 2012.

[6] K. Birman, G. Chockler, and R. van Renesse, “Toward a cloud computing research agenda,” *ACM SIGACT News*, vol. 40, no. 2, pp. 68–80, 2009.

[7] R. Amorim, D. B. Claro, D. Lopes, P. Albers, and A. Andrade, “Improving web service discovery by a functional and structural approach,” in *Proceedings of the 9th IEEE International Conference on Web Services (ICWS ’11)*, pp. 411–418, Washington, DC, USA, July 2011.

[8] W. Tan, Y. Fan, M. Zhou, and Z. Tian, “Data-driven service composition in enterprise soa solutions: a petri net approach,” *IEEE Transactions on Automation Science and Engineering*, vol. 7, no. 3, pp. 686–694, 2010.

[9] W. Liu, B. Liu, D. Sun et al., “Study on multi-task oriented services composition and optimisation with the “Multi-composition for each task” pattern in cloud manufacturing systems,” *International Journal of Computer Integrated Manufacturing*, vol. 26, no. 8, pp. 786–805, 2013.

[10] S. K. Bansal, A. Bansal, and M. B. Blake, “Trust-based dynamic web service composition using social network analysis,” in *Proceedings of the IEEE International Workshop on Business Applications of Social Network Analysis (BASNA ’10)*, pp. 1–8, Bangalore, India, December 2010.

[11] N. B. Mabrouk, S. Beauche, E. Kuznetsova et al., “QoS-aware service composition in dynamic service oriented environments,” in *Proceedings of the 10th ACM/IFIP/USENIX International Conference on Middleware*, p. 7, Springer, 2009.

[12] Y.-H. Shen and X.-H. Yang, “A self-optimizing QoS-aware service composition approach in a context sensitive environment,” *Journal of Zhejiang University: Science C*, vol. 12, no. 3, pp. 221–238, 2011.

[13] Y. Wei and M. B. Blake, “Service-oriented computing and cloud computing: challenges and opportunities,” *IEEE Internet Computing*, vol. 14, no. 6, pp. 72–75, 2010.

[14] F. Tao, D. Zhao, Y. Hu, and Z. Zhou, “Resource service composition and its optimal-selection based on particle swarm optimization in manufacturing grid system,” *IEEE Transactions on Industrial Informatics*, vol. 4, no. 4, pp. 315–327, 2008.

[15] M. Alrifai and T. Risse, “Combining global optimization with local selection for efficient QoS-aware service composition,” in *Proceedings of the 18th International Conference on World Wide Web*, pp. 881–890, ACM, 2009.

Hindawi Publishing Corporation
Journal of Applied Mathematics
Volume 2014, Article ID 327465, 10 pages
http://dx.doi.org/10.1155/2014/327465

Research Article

A Case Study on Formal Analysis of an Automated Guided Vehicle System

Jie Zhang,[1,2] Yuntao Peng,[1] William N. N. Hung,[3] Xiaojuan Li,[4] Jindong Tan,[2] and Zhiping Shi[4]

[1] *College of Information Science and Technology, Beijing University of Chemical Technology, Beijing, China*
[2] *University of Tennessee, Knoxville, TN, USA*
[3] *Synopsys Inc., Mountain View, CA, USA*
[4] *Beijing Engineering Research Center of High Reliable Embedded System, College of Information Engineering, Capital Normal University, Beijing, China*

Correspondence should be addressed to Jie Zhang; jzhang@mail.buct.edu.cn

Received 6 March 2014; Accepted 30 March 2014; Published 4 May 2014

Academic Editor: Xiaoyu Song

This paper considers a hybrid I/O automata model for an automated guided vehicle (AGV) system. A set of key properties of an AGV system are characterized for the correctness of the system. An abstract model is constructed from the hybrid automata model to simplify the proof of the constraints. The two models are equivalent in terms of bisimulation relation. We derive the constraints to ensure the correctness of the properties. We validate the system by analyzing the parameters of the constraints of the AGV system.

1. Introduction

Complex systems cannot be described by a pure discrete model or a continuous model [1–3]. Hybrid models have become increasingly popular in the last few decades as systems become increasingly complex. A hybrid system is a dynamic system with interacting continuous time triggered and discrete event triggered dynamics [1, 2, 4–6]. Many applications involve hybrid systems, such as embedded controllers [7], robotics [8, 9], mobile computing [10], and process control [11], in which high reliability is a requirement [4]. To model such a system, we need to describe and analyze it with the rigorous use of mathematics. An I/O automaton is used to model concurrent and distributed discrete event systems [12]. A hybrid input/output automaton (HIOA) [4] is a framework, which is developed by Lynch et al. and extended from hybrid automata for modeling complex hybrid systems. This is done by dividing the state variables of a HIOA into two sets, classified as internal variables and external variables, where the external variables include input variables and output variables. Discrete transitions and continuous trajectories can change the states of a system. An extremely important feature of the hybrid I/O automaton framework is that the hybrid system is divided into multiple modules. These modules are described so that the hybrid system can be modeled easily. The hybrid I/O automaton uses the external variables, input variables, and output variables to communicate among the automatons.

Automated guided vehicles (AGVs) are robots that move on the floor of a facility directed by a combination of software and sensor-based guidance systems. Earlier inventions on AGVs can be dated back to Barrett Electronics in 1953. One of the oldest publications on AGV can be found in [13]. In the past, AGVs were typically deployed to manufacturing facilities due to their efficiency, accuracy, and flexibility. Nowadays, AGVs are also used in warehouses, distribution centers and transshipment terminals, and so forth for repeated transportation tasks [14, 15]. The tracking path for the AGV can be designed as a circle, ellipse, sine wave, or other shapes such as arbitrary curves [16, 17]. The tracking trajectory is very important as many papers develop effective approaches to solve it, but our AGV is an example of applying HIOA modeling. Our modeling is inspired by [2]. But unlike [2] which uses a straight line orbit that can be approximated to one-dimension, we investigate a two-dimensional problem where an automated guided vehicle moves along a circular painted orbit.

The first contribution of this paper is the formal modeling of an automated guided vehicle system using hybrid I/O automata. The second contribution of this paper is a set of important constraints which are characterized to ensure the correctness of the properties of the vehicle system. In order to simplify the model, we abstract a model from the hybrid automata of the AGV and establish a bisimulation relation between the two automata.

This paper is organized as follows. In Section 2, an automated guided vehicle system is introduced. In Section 3, the HIOA framework is introduced. In Section 4, we present a HIOA model of the AGV system and abstract a model from HIOA model. We prove that the two models have a bisimulation relation. In Section 5, we extract the key properties and deduce the corresponding constraints to ensure the correctness of the properties. We analyze the parameters of the AGV system at the end of Section 6. Finally, we point out some directions for future work.

2. An Automated Guided Vehicle System

We introduce the structure and behavior of a vehicle. The vehicle consists of five components: the left wheels, the right wheels, chassis, sensor, and controller. Figure 1 shows a circular orbit tracking of our vehicle which is the focus of the remainder of this paper. The vehicle has two degrees of freedom. One is the velocity such that, at any time t, it can move forward with a speed of $v(t)$, with the restriction that $0 \leq v(t) \leq 10$ mph (miles per hour). The other degree of freedom is the circular movement of the vehicle such that at any time t the vehicle can rotate its body via the wheels with an angular speed of $-\pi \leq \omega(t) \leq \pi$ rad/s (radians per second). Ignoring the inertia of the vehicle, we assume that we can instantaneously change the velocity or angular speed. The sensor measures the displacement $e(t)$ between the center of the vehicle and the center of the track using an array of photodiodes. As the AGV passes over the track, the diode directly above the track generates more current than the other diodes. If the vehicle is close enough to the track, it will move forward. When the vehicle strays too far to the left, it will steer to the right; and when the vehicle strays too far to the right, it will steer to the left. The vehicle can be stopped at any time as long as it receives the control signal. If the vehicle is too far away from the track that it is difficult to follow the track, then it moves backward.

3. Hybrid I/O Automata Framework

In this section, we first introduce some basic notions about the model we use and then consider the definitions and theories of hybrid automata, hybrid I/O automata, and their operations [4]. More detailed discussion of the hybrid I/O automata can be found in [4].

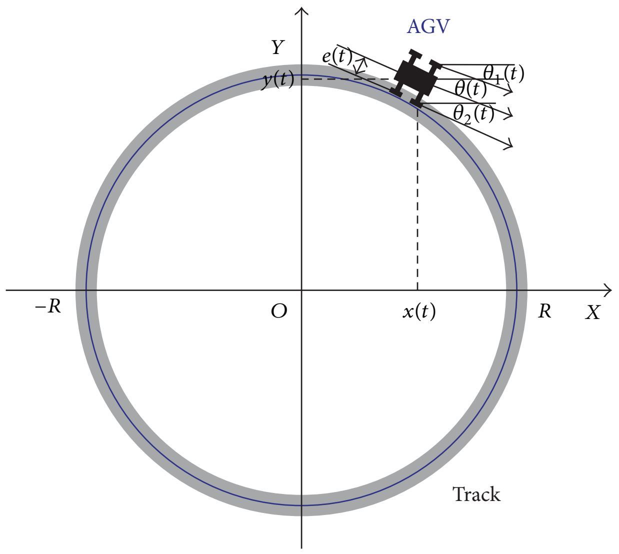


Figure 1: AGV tracking circular painted orbit.

3.1. Basic Notions. Hybrid behaviors, including discrete behavior, continuous behavior, and information flows into the system, are often described using static and dynamic variables, trajectories, and hybrid sequences. First, we introduce several basic notions involved in hybrid behavior. A location of the internal state of a system or a location of a connection between a component of a system and a component of another system can be represented as a variable, which may be static in type and denote a set of values of the variable, or dynamic in type and indicate a set of trajectories of the variable. A set of variables can be changed by discrete transitions, which are taken via discrete actions when they are enabled or by trajectories over a time interval. A hybrid sequence represents a series of changes that occur instantaneously along with the evolution of time and may be finite or infinite.

3.2. Hybrid Automata. As hybrid I/O automata are an extension of hybrid automata, we define the structure of hybrid automata first in order to describe the hybrid I/O automata. The definition of hybrid automata is given below, where $\triangleq$ denotes mathematical definition. For a more detailed description, see [4].

Definition 1. A hybrid automaton (HA) is an eight-tuple $H_M = (E_V, I_V, S, s_0, E_A, I_A, D_T, T)$, where

(i) E_V is a set of external variables,

(ii) I_V is a set of internal variables, and $V \triangleq E_V \cup I_V$ is the disjunction that represent all variables,

(iii) $S \subseteq \text{val}(I_V)$ is a set of states,

(iv) $s_0 \subseteq S$ is a nonempty set of initial states,

(v) E_A is a set of external actions,

(vi) I_A is a set of internal actions, and $A \triangleq E_A \cup I_A$ is the union of E_A and I_A,

(vii) $D_T \subseteq S \times A \times S$ is a set of discrete transitions,

(viii) T is a set of trajectories for V. For every $\tau \in T$ and $t \in \mathrm{dom}(\tau)$ (domain of τ), we have $\tau(t)\lceil I_V \in S$, where $\tau(t)\lceil I_v$ is the restriction of $\tau(t)$ to I_V; that is, the function g with $\mathrm{dom}(g) = \mathrm{dom}(\tau) \cap I_v$ such that $g(t) = \tau(t)$. We require the following axioms:

(A1) for all $\tau \in T$ for all $\tau' \leq \tau$ $\tau' \in T$;

(A2) for all $\tau \in T$ for all $t \in \mathrm{dom}(\tau)$ $\tau \rhd t \in T$ where $\tau \rhd t$ denotes $((\tau\lceil[t, +\infty)) - t)$;

(A3) suppose $\tau_0, \tau_1, \tau_2, \ldots, \tau_i, \ldots$ is a sequence of trajectories in T; if τ_i is closed and $\tau_i \cdot l\text{state} = \tau_{i+1} \cdot f\text{state}$, where $i \in \mathbb{N}$, τ_i is not the last trajectory of the hybrid sequence, then $\tau_0 \frown \tau_1 \frown \tau_2 \cdots \in T$, where $\tau \frown \tau' \triangleq \tau \cup (\tau'\lceil(0, \infty) + \tau \cdot l\text{time})$.

The execution fragment of a hybrid automaton is a hybrid sequence $\alpha = \tau_0 a_1 \tau_1 a_2, \ldots, \tau_i a_{i+1}, \ldots$, where $\tau_i \in T$, where i is a nonnegative integer and T is defined in Definition 1; and if τ_i is not the last trajectory, then $\tau_i \cdot l\text{state} \xrightarrow{a_{i+1}} \tau_{i+1} \cdot f\text{state}$, where $l$state represents the last state and $f$state denotes the first state. Any input trajectory of the composition can be accepted by the composition, and we say that the components of the composition are strongly compatible HIOAs. Trace is the external behavior of a hybrid I/O automaton. Concatenation represents two hybrid sequences linked together. Let and be hybrid sequences and closed, with the concatenation being denoted by $\alpha \frown \alpha' \triangleq \mathrm{init}(\alpha)(\mathrm{last}(\alpha) \frown \mathrm{head}(\alpha'))\mathrm{tail}(\alpha')$.

3.3. Hybrid I/O Automata.

We described the hybrid automata above. Here, we present the behavior and structure of a HIOA. A HIOA is used to model a complex hybrid system. The discrete state of the controller can be modeled by control modes, represented as internal variables. Each mode observes an invariant condition. The internal variables can be changed in two ways: in a discrete transition or in a continuous trajectory. External variables, including input variables and output variables, are used to exchange information between two automatons. Here is the definition of a hybrid input/output automaton. For a more detailed description, see [4].

Definition 2. A hybrid I/O automaton (HIOA) is a five-tuple $A_M = (H_M, U, Y, I, O)$, where

(i) $H_M = (E_V, I_V, S, s_0, E_A, I_A, D_T, T)$ is a hybrid automaton,

(ii) $U \subseteq E_V$ is a set of input variables,

(iii) $Y \triangleq E_V \setminus U$ is a set of output variables,

(iv) I is a set of input actions,

(v) O is a set of output actions,

(vi) the following axioms are satisfied:

(A1) for all $x \in S$ for all $\alpha \in I$ $\exists x' \in S$ such that $x \xrightarrow{a} x'$,

(A2) let $\mathrm{trajs}(U)$ denote the set of all trajectories for U, for all $x \in S$ for all $v \in \mathrm{trajs}(U)$ $\exists \tau \in T$ such that $\tau \cdot f\text{state} = x, \tau \downarrow U \leq v$, and either

(a) $\tau \downarrow U = v$, or

(b) τ is closed and some $l \in L$ is enabled in $\tau \cdot l\text{state}$,

where $g = \tau \downarrow U$ represents $\mathrm{dom}(g) = \mathrm{dom}(\tau)$ such that, for all $c \in \mathrm{dom}(g)$ has $g(c) = \tau(c)\lceil U$.

We further define

(i) $Z \triangleq I_V \cup Y$ is a set of variables that are locally controlled, and

(ii) $L \triangleq I_A \cup O$ is a set of actions that are locally controlled.

Typically, it is difficult to model a complex system in one shot. HIOA can decompose a hybrid system into multiple components, model the modules as HIOAs, respectively, and then compose them in the end. We introduce a very important operation to compose two HIOAs, denoted as symbol $\|$. For the proof of Theorem 3 and Lemma 4, see [4].

Theorem 3. *$A_{M1} \parallel A_{M2}$ is a hybrid I/O automaton when A_{M1} and A_{M2} are strongly compatible hybrid I/O automata and $U_1 \cap Y_2 = \emptyset$.*

Another important operation is hiding external variables in HIOA. Suppose $E_V \subseteq E_{V_A}$, $B_M = \mathrm{VarHide}(E_V, A_M)$, $E_{V_B} = E_{V_A} - E_V$, and $T_{B_M} = T_{A_M} \downarrow (V_{A_M} - E_V)$.

Lemma 4. *If A_M is a HIOA and $E_V \subseteq E_{V_{A_M}}$, then $\mathrm{VarHide}(E_V, A_M)$ is a HIOA.*

Definition 5 (simulation relations). For all states x_{A_M} and x_{B_M} of A_M and B_M, given two comparable HIOAs, from A_M to B_M there exists a simulation relation $R_S \subseteq S_{A_M} \times S_{B_M}$ (denoted as $A_M R_S B_M$) when the following three conditions are met:

(i) knowing that $x_{A_M} \in s_{0_{A_M}}$ and suppose there exists a state $x_{B_M} \in s_{0_{B_M}}$ such that $x_{A_M} R_S x_{B_M}$, where $s_{0_{A_M}}$ is the set of initial states of A_M and $s_{0_{B_M}}$ is the set of initial states of B_M;

(ii) suppose $x_{A_M} R_S x_{B_M}$ and an execution fragment of A_M; execution fragment $\alpha = \tau_0 a_1 \tau_1 a_2, \ldots, \tau_i a_{i+1}, \ldots$, meets $\alpha \cdot f\text{state} = x_{A_M}$; there exists a closed execution fragment β in B_M that meets $\beta \cdot f\text{state} = x_{B_M}$, $\mathrm{trace}(\beta) = \mathrm{trace}(\alpha)$, and $\alpha \cdot f\text{state} R \beta \cdot f\text{state}$;

(iii) suppose $x_{A_M} R_S x_{B_M}$ and an execution fragment of $A_M \alpha = \tau_0$ has $\alpha \cdot f\text{state} = x_{B_M}$; there exists a closed execution fragment β in B_M that meets $\beta \cdot f\text{state} = x_{B_M}$, $\mathrm{trace}(\beta) = \mathrm{trace}(\alpha)$, and $\alpha \cdot l\text{state} R \beta \cdot l\text{state}$.

Corollary 6. *Given two comparable HAs A_M and B_M, and a simulation from A_M to B_M denoted as $A_M R_s B_M$, then $\mathrm{traces}_{A_M} \subseteq \mathrm{traces}_{B_M}$.*

The proof refers to [4]. According to [18, 19], we define a bisimulation as follows.

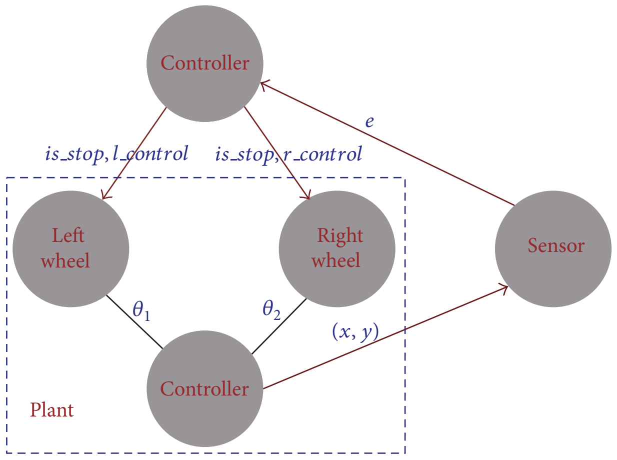


Figure 2: Network of hybrid automata for an AGV.

Definition 7 (bisimulation). Given two comparable HIOAs A_M and B_M, for all pairs (p, q) among all reachable states of A_M and B_M, p in A_M, and q in B_M. If all reachable states p^* in A_M have $p \xrightarrow{a} p^*$, this implies the existence of a state q^* in B_M such that $p \xrightarrow{a} p^*$. At the same time, all reachable states q^* in B_M have $p \xrightarrow{a} p^*$, implying that there exists a state q^* in A_M such that $p \xrightarrow{a} p^*$. Under these circumstances, we say that A_M and B_M have a bisimulation relation.

4. Modeling the AGV System

We model the AGV system using HIOA. Inspired by [2], the AGV system is modeled as a network of hybrid automata as shown in Figure 2. The model consists of five parts: chassis, left wheels, right wheels, sensor, and controller, respectively. The five components communicate via shared variables. In Figure 2, variables θ_1 and θ_2 are the angles of the left and right wheels relative to the x-axis positive direction of globe coordinate, respectively. Variables x and y represent the chassis coordinates with respect to the global coordinate frame. Variable e is the distance $e(t)$ from which the center of the AGV deviates from the center of the track at time t. The variable is used to communicate between the sensor and the controller. The controller receives the variable e, sends control signals to the left wheel and the right wheel, and then changes the mode of the AGV.

XOY is the global coordinate frame. v is the forward velocity of the car. t_{sample} is the sampling time. ω is the angular speed of the vehicle. $e(t)$ is the displacement of the center of the vehicle from the track at time t. ε_1 is the threshold indicating that the AGV is close enough to the center of the track that the AGV can move straight ahead in a forward mode. ε_2 is the threshold indicating that there is too great a distance between the center of the AGV and the center of track, and that the vehicle must therefore be steered to the other side. ε_3 is the threshold denoting that the vehicle has strayed so far from the center of track that the vehicle is in an unsafe condition and must be moved back via switching to the back mode. α is the maximum angle of vehicle velocity direction to the tangential direction of the center point on the track, where $0 \le \alpha \le \pi/2$. θ is the angle of the vehicle velocity direction to the x-axis positive direction. η is the angle of the vehicle velocity direction to the tangential direction of the center point on the track, where $-\alpha \le \eta \le \alpha$. R is the radius of the track.

The AGV system is decomposed into five components and modeled as hybrid automata: chassis, LWheel, RWheel, sensor, and controller, respectively.

4.1. Component Chassis. The chassis secures the position of each component. The state is composed of three state variables: $\langle x, y, \theta \rangle$ where x is the x-coordinate of the center of the vehicle; y is the y-coordinate of the center of the vehicle; and θ is the angle of the vehicle velocity direction to the x-axis positive direction. We use differential algebraic equations (DAEs) to describe the dynamic of the chassis. Initially, we ensure that the vehicle moves forward, and the initial condition is

$$\eta \in [-\alpha, \alpha] \wedge |e(t)| \le \varepsilon_1. \tag{1}$$

From Figure 2, the chassis secures the wheels. Hence the left wheels, right wheels, and the chassis have the same angles. We obtain the following algebraic equation:

$$\theta_1 = \theta_2 = \theta. \tag{2}$$

4.2. Component LWheel. We model the behavior of left wheels as the hybrid automaton LWheel. The left wheel has external variables: x_1, which gives the x-coordinate of the left wheel; y_1, which gives the y-coordinate of the right wheel; and θ_1, which is the angle of the moving direction of the left wheel to the x-axis positive direction. The types of these variables are real. This hybrid automaton model has no actions or discrete transitions, just satisfied trajectories. It communicates with the controller via the Boolean variable l_control and is_stop. We obtain differential equations for the left wheel as follows:

$$\begin{aligned}\frac{dx_1}{dt} &= \textbf{if } \text{stop } \textbf{then } 0 \\ &\quad \textbf{else if } l\text{\_control } \textbf{then } v\cos\ \theta_1 \\ &\quad \textbf{else } -v\cos\theta_1 \\ \frac{dy_1}{dt} &= \textbf{if } \text{stop } \textbf{then } 0 \\ &\quad \textbf{else if } l\text{\_control } \textbf{then } v\sin\ \theta_1 \\ &\quad \textbf{else } -v\sin\theta_1.\end{aligned} \tag{3}$$

4.3. Component RWheel. Since the left wheels and right wheels are symmetrical, we omit the description of the right wheels.

4.4. Component Sensor. We model the behavior of the *sensor* as the hybrid automaton sensor, whose output at time t, for all t $e(t) = f(x(t), y(t))$, gives the center position of the AGV relative to the center of the track, shown in Figure 1. The sensor communicates with the controller through the variable e, which equals $e(t)$. Since the hybrid automaton sensor has no internal variables, and there are neither actions nor discrete transitions, only the following algebraic equation is met for the trajectories of the sensor:

$$\text{sensor} = e(t) = f(x(t), y(t)). \tag{4}$$

4.5. Component Controller. The controller can be divided into two levels. The supervisory controller determines the structure of the mode transition and guards the enabled transitions. The low-level controller determines the time-based inputs to the system. We are modeling the behavior of the controller as a hybrid automaton *controller*, whose input is the sensor value and the output the control signals to the left and right wheels that determine the operation of the wheels. There is a clock built into the controller for measuring the time interval since the last sampling. We use the variable c to represent this clock. A clock can be modeled as a first-order differential equation, and the clock variable c is defined as follows:

$$c = k \cdot t, \tag{5}$$

where k is the rate of the clock, t is the variable of time, and $k = d[c(t)]/dt$. In our model, the value of k can be a constant 1.

The controller has a variable e, which gets its value from the sensor. There are two variables recording the value of the sensor: variable new_sample, used to record the latest sample value, and variable sample, used to record the last sample value. In order to ensure that the vehicle moves forward, the initial states should satisfy:

$$c = 0 \land \text{sample} \in [-\epsilon_1, \epsilon_1]. \tag{6}$$

We define a transition as occurring when a guard in an outgoing transition from the current state becomes enabled. This control logic is captured in the mode transitions. The outputs are the pure signals is_stop, forward, and backward. There are three Boolean variables recording the outputs, is_stop, l_control, and r_control, respectively. We use an asterisk * to represent the next sample value. When the internal action clock transitions is taken, each state transition that is enabled will be taken:

$$c = t_{\text{sample}} \rightsquigarrow \begin{cases} c^* = 0 \\ \text{sample}^* = \text{new\_sample}, \end{cases} \tag{7}$$

where $\rightsquigarrow$ denotes the event trigger.

For trajectories we require that

$$\text{sample}(t_1) = \text{sample}(t_2), \tag{8}$$

for all time t_1, t_2 between clock transitions; that is, $t_1, t_2 \in [m \cdot t_{\text{sample}}, (m+1) \cdot t_{\text{sample}})$ for all $m \in \mathbb{N}$.

The new_sample will record the new value from the sensor:

$$\text{new\_sample} = e(t). \tag{9}$$

For every state, the following equations must hold:

$$\begin{aligned} &\frac{dc}{dt} = 1 \\ &l\text{\_control} = \textbf{if } \text{sample} \in [-\epsilon_3, \epsilon_3] \textbf{ then } \text{true} \\ &\qquad \textbf{else } \text{false} \\ &r\text{\_control} = \textbf{if } \text{sample} \in [-\epsilon_3, \epsilon_3] \textbf{ then } \text{true} \\ &\qquad \textbf{else } \text{false} \\ &\text{is\_stop} = \textbf{if } \text{stop } \textbf{then } \text{true} \\ &\qquad \textbf{else } \text{false}. \end{aligned} \tag{10}$$

The control logic determines the change in the state of the controller. Our AGV is running on the circular track. Since the circle is symmetrical, it suffices for us to just consider the situation of the first quadrant. The refinement of the mode gives the dynamic behavior of the output as a function of the input. We know that the displacement $e(t)$ is the function of x and y and that the control logic guards the transitions whether enabled or not. They are as follows:

$$\begin{aligned} &\text{forward} \Longrightarrow \neg\text{stop} \land -\epsilon_1 \le e(t) \le \epsilon_1 \\ &\Longrightarrow \begin{cases} x' = v\cos\theta, \\ y' = v\sin\theta, \\ \theta' = 0 \end{cases} \\ &\text{right} \Longrightarrow \neg\text{stop} \land \epsilon_3 \ge e(t) > \epsilon_2 \\ &\Longrightarrow \begin{cases} x' = v\cos\theta, \\ y' = v\sin\theta, \\ \theta' = -\omega \end{cases} \\ &\text{left} \Longrightarrow \neg\text{stop} \land -\epsilon_3 \le e(t) < -\epsilon_2 \\ &\Longrightarrow \begin{cases} x' = v\cos\theta, \\ y' = v\sin\theta, \\ \theta' = \omega \end{cases} \\ &\text{back} \Longrightarrow \neg\text{stop} \land e(t) < -\epsilon_3 \lor e(t) > \epsilon_3 \\ &\Longrightarrow \begin{cases} x' = -v\cos\theta, \\ y' = -v\sin\theta, \\ \theta' = 0 \end{cases} \\ &\text{stop} \Longrightarrow \begin{cases} x' = 0, \\ y' = 0, \\ \theta' = 0 \end{cases} \\ &\text{maintain} \Longrightarrow \begin{cases} -\epsilon_2 \le e(t) < -\epsilon_1 \quad \text{or} \\ \epsilon_1 < e(t) \le \epsilon_2. \end{cases} \end{aligned} \tag{11}$$

4.6. Composition. Since the left wheels, the right wheels, and the chassis have no output, they cannot be regarded as hybrid I/O automata. Since θ_1 and θ_2 are the internal variables of wheels, we are modeling the three components as the hybrid I/O automaton Plant by hiding these variables. In our model, Plant, is_stop, l_control, and r_control are inputs, and x and y are outputs:

$$\text{Plant} = \text{VarHide}\left(\{\theta_1, \theta_2\}, (\text{Chassis} \| \text{LWheel} \| \text{RWheel})\right). \quad (12)$$

Likewise, the sensor can be regarded as a hybrid I/O automaton, for which the inputs are x and y, and the output is e. The controller can also be viewed as a hybrid I/O automaton for which the input is e, and the outputs are is_stop, l_control, and r_control. According to Theorem 3 and Lemma 4, all of the components of the system are HIOAs and the composition also an HIOA. We have obtained a complete hybrid I/O automaton of the AGV system by hiding the external variables:

$$H_M = \text{VarHide}\left(\{x, y, e, \text{is\_stop}, l\text{\_control}, r\text{\_control}\}, (\text{Plant} \| \text{Sensor} \| \text{Controller})\right). \quad (13)$$

4.7. Abstraction. We expect that the AGV is always moving forward and never moves backward and use (14) to describe this situation. We select the appropriate threshold and ensure that the vehicle moves in the way we expect by specifying parameter constraints for all reachable states of the hybrid I/O automata H_M:

$$|\text{sample}| \le \epsilon_3. \quad (14)$$

In addition, we hope that the forward mode occurs infinitely often:

$$\text{GF}\left(c = 0 \land |\text{sample}| \le \epsilon_1\right). \quad (15)$$

In order to simplify the model, we abstract a model A_M from the previous model H_M. Then, we use model A_M instead of H_M. We find the constraints we need from model A_M to guarantee the correctness of the properties that we expect. Here, we simplify the model in several ways, as follows.

Based on (8), we know that the value of the variable sample remains unchanged during the interval after the current sampling and before the next sampling. Therefore we can easily prove that (14) is satisfied. We cannot consider the influence of the clock variable c. Furthermore, we assume that the vehicle is at the initial state at the time 0:

$$c = 0 \Longrightarrow |e(t)| \le \epsilon_3. \quad (16)$$

We find that the variables new_sample and sample are ruled out in our abstract model. Now, we use the refinements of the five modes of AGV to describe the dynamic behavior of an AGV. The formulas of the five modes are given as $\varphi_{\text{forward}}, \varphi_{\text{right}}, \varphi_{\text{left}}, \varphi_{\text{back}}$, and φ_{stop}, respectively; φ_{step} is the disjunction of the five:

$$\begin{aligned}
\varphi_{\text{step}} &\triangleq \varphi_{\text{right}} \lor \varphi_{\text{left}} \\
&\quad \lor \varphi_{\text{forward}} \lor \varphi_{\text{back}} \lor \varphi_{\text{stop}} \\
\varphi_{\text{forward}} &\triangleq \begin{cases} \neg\text{stop}, \\ |e(t)| \le \epsilon_1, \\ x^* = x + v\cos\theta t_{\text{sample}}, \\ y^* = y + v\sin\theta t_{\text{sample}}, \\ \theta^* = \theta \end{cases} \\
\varphi_{\text{right}} &\triangleq \begin{cases} \neg\text{stop}, \\ e(t) > \epsilon_2, \\ x^* = x + v\cos\theta t_{\text{sample}}, \\ y^* = y + v\sin\theta t_{\text{sample}}, \\ \theta^* = \theta - \omega t_{\text{sample}} \end{cases} \\
\varphi_{\text{left}} &\triangleq \begin{cases} \neg\text{stop}, \\ e(t) < -\epsilon_2, \\ x^* = x + v\cos\theta t_{\text{sample}}, \\ y^* = y + v\sin\theta t_{\text{sample}}, \\ \theta^* = \theta + \omega t_{\text{sample}} \end{cases} \\
\varphi_{\text{back}} &\triangleq \begin{cases} \neg\text{stop}, \\ |e(t)| > \epsilon_3, \\ x^* = x - v\cos\theta t_{\text{sample}}, \\ y^* = y - v\sin\theta t_{\text{sample}}, \\ \theta^* = \theta \end{cases} \\
\varphi_{\text{stop}} &\triangleq \begin{cases} \text{stop}, \\ x^* = x, \\ y^* = y, \\ \theta^* = \theta. \end{cases}
\end{aligned} \quad (17)$$

Now we get the abstract model of the AGV system. Since the abstract model A omits the time variable, it is simpler than the original model. We will derive and verify the properties using the abstract model. There are two kinds of typical errors in formal verification, one is true error, where errors exist in the physical system, but the result of formal verification is correct. The reason of first kind of error is because we abstract a model from our original model, and the details we omitted may lead to the errors of original model being omitted, so we get a passing proof. The other is false error which do not exist in the physical system but the result of formal verification is incorrect. The reason is that abstract model omitted the details of original model. The abstract model cannot express the original system due to lack of information from the original system, and then the result of formal verification is incorrect.

In order to ensure the two kinds of errors never occur, we prove the original model H_M and the abstract model A_M have a bisimulation equivalence relationship.

Lemma 8. *Let S_K be a set of all reachable states of H_M, for all $s \in S_R$; one has*

$$c = 0 \Longrightarrow [|\,sample\,| \le \epsilon_2 \Longleftrightarrow |e(t)| \le \epsilon_2] \quad (18)$$

Proof. Use (14) and (15) to prove Lemma 8. □

Theorem 9. *The two comparable HIOAs H_M and A_M have a bisimulation relation.*

Proof. Owing to the limitation of space, we do not provide a detailed proof of Theorem 9, but the key step will be given. H_M and A_M satisfy the following condition:

$$c_{H_M} = 0 \wedge \theta_{H_M} = \theta_{A_M} \wedge x_{H_M} = x_{A_M} \wedge y_{H_M} = y_{A_M}. \quad (19)$$

For all state pairs (p, q) among all reachable states of H_M and A_M, $p \in H_M$ and $q \in A_M$, if states of state pair (p, q) hold the weakest condition of labeled transition system respectively, we say the pair (p, q) is bisimulation equivalent. If each initial state of H_M bisimulates an initial state of A_M, and there exists an execution fragment from p to p^*, where p^* in H_M has $p \rightarrow p^*$, implying the existence of a transition according to the transition predicate φ_{step} of A_M from q to q^* in A_M, such that $q \Rightarrow q^*$. At the same time, there exists a transition according to the transition predicate φ_{step} of A_M from q to q^*, where q^* in A_M has $q \Rightarrow q^*$, implying the existence of an execution fragment from p to p^*, where p^* in H_M, such that $p \rightarrow p^*$. We can then use Definition 5, Corollary 6, and Definition 7 to prove Theorem 9. □

5. Correctness

5.1. The Desired Properties of A_M. For a system, we often hope that bad things will never happen, a situation called safety, that good things will eventually happen, and that they will happen infinitely often, a situation called fairness. We express the properties via invariants. For our system, we expect the displacement from the center of the AGV to the center of the track to never be larger than the threshold ε_3, and never be less than the threshold $-\varepsilon_3$. At the same time, we ensure that η lies in the interval $[-\alpha, \alpha]$.

Property 1. The vehicle always moves forward and never moves backward. It can be described as

$$\varphi_{\text{safety}} \triangleq \eta \in [-\alpha, \alpha] \wedge |e(t)| \le \varepsilon_3. \quad (20)$$

Property 2. The vehicle moves forward infinitely often. It can be described using the temporal logic formula

$$\varphi_{\text{fairness}} \triangleq \text{GF}\,|e(t)| \le \varepsilon_1. \quad (21)$$

Lemma 10. *If φ_{safety} is an invariant of A_M and formula $\varphi_{fairness}$ holds for A_M, then* (14) *is an invariant of H_M and* (15) *holds for H_M.*

Proof. Use Lemma 8, Theorem 9, and (8) to prove Lemma 10. □

5.2. Parameter Constraints of the AGV System. In this section, we will give several parameter constraints for our AGV system. They are indispensable to guaranteeing the correctness of the properties of safety (20) and fairness (21). We define them in (22).

Parameter Constraints. Consider the following

$$\begin{aligned}
\varphi_1 &\triangleq \left(v\cos\alpha t_{\text{sample}}\right)^2 + \left(R + \varepsilon_2 + v\sin\alpha t_{\text{sample}}\right)^2 \\
&\le (\varepsilon_3 + R) \\
\varphi_2 &\triangleq (R - \varepsilon_2)^2 + v^2 t_{\text{sample}}^2 - 2(R - \varepsilon_2)\,v t_{\text{sample}} \sin\alpha \\
&\ge (R - \varepsilon_3)^2 \\
\varphi_3 &\triangleq \omega t_{\text{sample}} \le \alpha \\
\varphi_4 &\triangleq (R - \varepsilon_3)^2 \\
&\le \left(v\cos\alpha t_{\text{sample}}\right)^2 + \left(R - \varepsilon_2 + v\sin\alpha t_{\text{sample}}\right)^2 \\
&\le (R + \varepsilon_1)^2.
\end{aligned} \quad (22)$$

Theorem 11. *If φ_1, φ_2, and φ_3 are met, then the φ_{safety} property is an invariant of A_M; that is,*

$$\varphi_{safety} \wedge \varphi_1 \wedge \varphi_2 \wedge \varphi_3 \Longrightarrow \varphi^*_{safety}. \quad (23)$$

Proof. In the first step, we prove that $\varphi_{\text{safety}} \wedge \varphi_1 \Rightarrow 0 \le e^*(t) \le \varepsilon_3$ holds. Since the circle is symmetrical, we only need to consider the situation of the first quadrant. In order to guarantee that $|e(t)| \le \varepsilon_3$ is met in all cases of outside the circle track, we consider the most extreme case of the outside of the circle. First of all, suppose that the vehicle moves on the outside of the circle shown in Figure 3. The vehicle is very close to point A at the time of the current sampling, and $e(t) \le \varepsilon_2$. The vehicle then moves forward to B with $\eta = \alpha$ at the next sampling; the $e(t)$ reaches the largest displacement. We use φ_1 to illustrate that $0 < e(t) \le \varepsilon_3$ holds for A_M. The derivations in (24) show that $e^*(t) \le \varepsilon_3$.

Deriving from φ_1. Consider the following:

$$\begin{gathered}
\left(v\cos\alpha t_{\text{sample}}\right)^2 + \left(R + \varepsilon_2 + v\sin\alpha t_{\text{sample}}\right)^2 \le (\varepsilon_3 + R)^2 \\
\sqrt{\left(v\cos\alpha t_{\text{sample}}\right)^2 + \left(R + \varepsilon_2 + v\sin\alpha t_{\text{sample}}\right)^2} \le (\varepsilon_3 + R) \\
\sqrt{\left(v\cos\alpha t_{\text{sample}}\right)^2 + \left(R + \varepsilon_2 + v\sin\alpha t_{\text{sample}}\right)^2} - R \le \varepsilon_3 \\
\sqrt{|BD|^2 + |OD|^2} - R \le \varepsilon_3 \quad \text{(use the Pythagorean theorem)} \\
\sqrt{|BD|^2} - R \le \varepsilon_3 \\
|OB| - R \le \varepsilon_3 \\
e^*(t) \le \varepsilon_3.
\end{gathered} \quad (24)$$

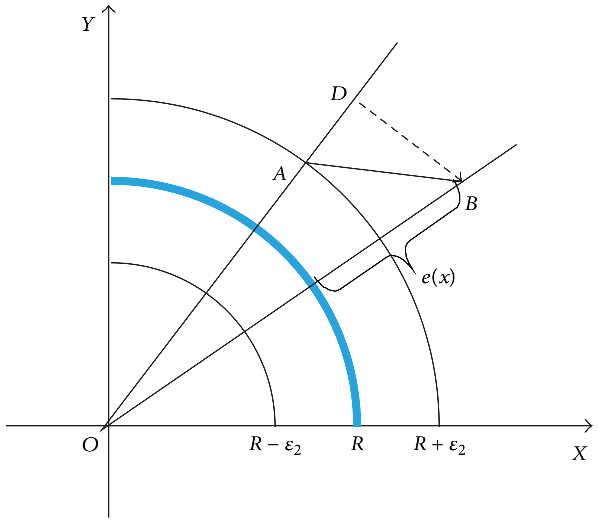


FIGURE 3: Illustration of the need for φ_1.

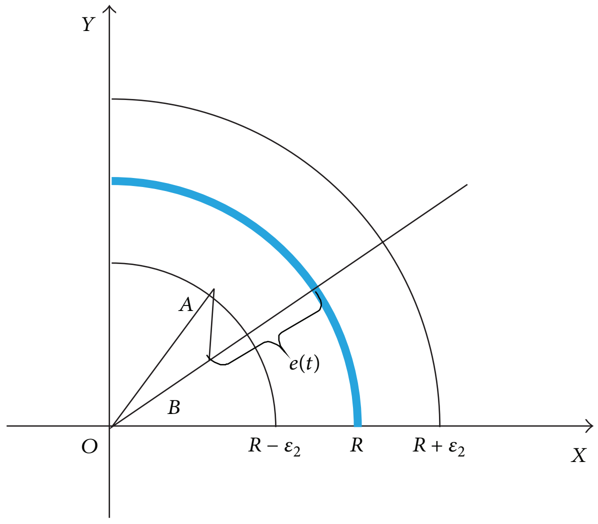


FIGURE 4: Illustration of the need for φ_2.

Since the vehicle moves on the outside of the circle, $e(t)^* > 0$. Therefore, $\varphi_{\text{safety}} \wedge \varphi_1 \Rightarrow 0 < e^*(t) \le \varepsilon_3$.

In the second step, we prove that $\varphi_{\text{safety}} \wedge \varphi_2 \Rightarrow -\varepsilon_3 \le e^*(t) \le 0$ holds. In order to guarantee that $|e(t)| \le \varepsilon_3$ in all cases inside the track of the circle, we consider the most extreme case inside. First of all, suppose that the vehicle moves on the inside of the circle shown in Figure 4. The vehicle is very close to point A at the time of the current sampling, with $e(t) > -\varepsilon_2$, and then moves forward to B with $\eta = -\alpha$ at the next sampling, where the vehicle reaches the farthest to the track. We use φ_2 to illustrate that $-\varphi_3 \le e(t) < 0$ holds for A_M. The derivations in (25) show that $e^*(t) \ge -\varepsilon_3$.

Deriving from φ_2. Consider the following:

$$
\begin{gathered}
(R-\varepsilon_2)^2 + v^2 t_{\text{sample}}^2 - 2(R-\varepsilon_2) v t_{\text{sample}} \sin\alpha \ge (R-\varepsilon_3)^2 \\
\sqrt{(R-\varepsilon_2)^2 + v^2 t_{\text{sample}}^2 - 2(R-\varepsilon_2) v |\sin(-\alpha)| t_{\text{sample}}} \\
\ge R-\varepsilon_3 \\
\sqrt{|OA|^2 + |AB|^2 - 2|OA\|AB\| \sin(-\alpha)|} \ge R-\varepsilon_3 \\
\sqrt{|OA|^2 + |AB|^2 - 2|OA|\,|AB| \cos\left(\frac{\pi}{2}-\alpha\right)} \ge R-\varepsilon_3 \\
\text{(use the Law of cosines)} \\
|OB| \ge R-\varepsilon_3 \\
OB - R \ge -\varepsilon_3 \\
e^*(t) \ge -\varepsilon_3.
\end{gathered}
\tag{25}
$$

Since the vehicle moves on the inside of the circle, $e^*(t) < 0$. Therefore, $\varphi_{\text{safety}} \wedge \varphi_2 \Rightarrow -\varepsilon_3 \le e^*(t) \le 0$.

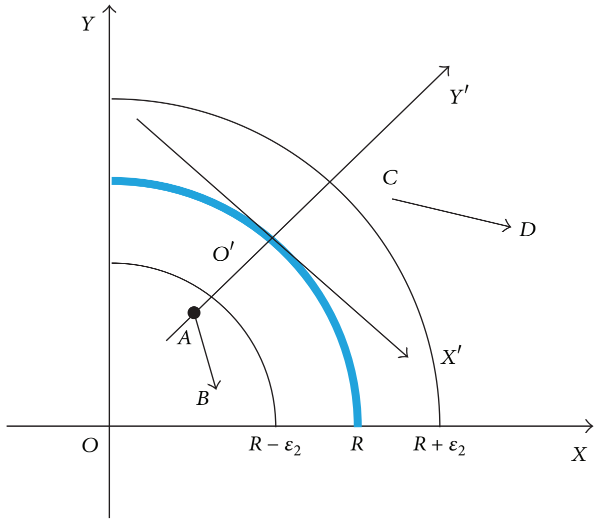


FIGURE 5: Illustration of the need for φ_3.

In the third step, we prove that $\varphi_{\text{safety}} \wedge \varphi_3 \Rightarrow \eta^* \in [-\alpha, \alpha]$ holds. Constraint $\omega t_{\text{sample}} \le \alpha$ is required to guarantee that η is always in the interval $[-\alpha, \alpha]$. First of all, we consider the scenario shown in Figure 5. We build a coordinate frame $X'O'Y'$ shown in Figure 5. If the vehicle reaches point A in the current sampling, the vehicle will steer to the left. The angle $\eta < 0$ (the angle between $\overrightarrow{AB}$ and $\overrightarrow{O'X'}$), relative to the coordinate frame $X'O'Y'$, is $\eta^* = \eta + \omega t_{\text{sample}} < \omega t_{\text{sample}}$ at the next sampling. If $\omega t_{\text{sample}} \le \alpha$, then $\eta^* = \eta + \omega t_{\text{sample}} < \alpha$. If the vehicle reaches point C in the current sampling, the vehicle will steer to the left. The angle $\eta > 0$ (the angle between $\overrightarrow{CD}$ and $\overrightarrow{O'X'}$), relative to the coordinate frame $X'O'Y'$, is $\eta^* = \eta - \omega t_{\text{sample}} > -\omega t_{\text{sample}}$ at the next sampling. If $\omega t_{\text{sample}} \le \alpha$, then $-\omega t_{\text{sample}} \ge -\alpha$, and we will get $\eta^* = \eta - \omega t_{\text{sample}} > -\alpha$. We have proved that $\eta^* \in [-\alpha, \alpha]$.

Therefore, $\varphi_{\text{safety}} \wedge \varphi_1 \wedge \varphi_2 \wedge \varphi_3 \Rightarrow \varphi^*_{\text{safety}}$ is proved. □

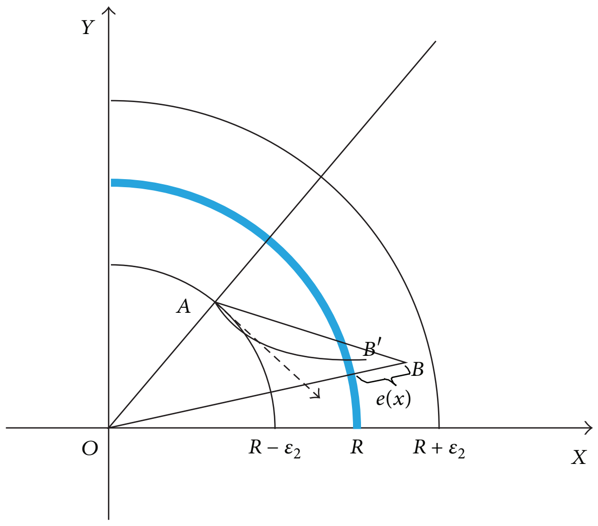


FIGURE 6: Illustration of the need for φ_4.

Theorem 12. *If φ_4 holds for A_M, then the property of $\varphi_{fairness}$ is an invariance of A_M; that is, $\varphi_{fairness} \wedge \varphi_4 \Rightarrow \varphi^*_{fairness}$.*

Proof. In order to ensure that the vehicle moves forward infinitely often, we avoid the situation of always steering to left after steering to right, and steering to right after steering to left. We consider the scenario shown in Figure 6. The center of the vehicle is very close to point A in the current sampling, and $e(t) < \varepsilon_2$, with the velocity direction approximate parallel to the direction of the tangent of point A, shown as a dashed line. The vehicle steers to the left, moves along the arc $\overset{\frown}{AB}$, and we look at $\overset{\frown}{AB}$ as a straight line AB. Suppose that $\eta = \alpha$, $e(t) < \varepsilon_1$ at the next sampling and that the vehicle switches to the forward mode. From φ_4, we can derive the following:

$$\begin{aligned} R - \varepsilon_1 \le |OB| \le R + \varepsilon_1 \;(R > \varepsilon_3 > \varepsilon_2 > \varepsilon_1) \\ -\varepsilon_1 \le |OB| - R \le -\varepsilon_1 \\ -\varepsilon_1 \le e^*(t) \le -\varepsilon_1 \\ |e^*(t)| \le -\varepsilon_1. \end{aligned} \tag{26}$$

□

6. Analysis of Constraints

In this section, we analyze the parameters of our AGV system. We rewrite the constraints as shown in (27).

Rewrite Constraints. Consider the following:

$$\begin{aligned} \varphi_1 &\triangleq v^2 t^2_{\text{sample}} + \varepsilon_2^2 + 2\varepsilon_2 v \sin\alpha t_{\text{sample}} \\ &\le \varepsilon_3^2 + 2R\left(\varepsilon_3 - \varepsilon_2 - v\sin\alpha t_{\text{sample}}\right) \\ \varphi_2 &\triangleq \left(R - \varepsilon_2 - v t_{\text{sample}}\right)^2 + 2\left(R - \varepsilon_2\right) v t_{\text{sample}} \left(1 - \sin\alpha\right) \\ &\ge \left(R - \varepsilon_3\right)^2 \\ \varphi_3 &\triangleq \omega t_{\text{sample}} \le \alpha. \end{aligned} \tag{27}$$

We assume that the value range of v is from $v_{\min}$ to $v_{\max}$, t_{sample} is from $t_{\min}$ to $t_{\max}$, α is from $\alpha_{\min}$ to $\alpha_{\max}$, ε_1 is from $\varepsilon_{1_{\min}}$ to $\varepsilon_{1_{\max}}$, ε_2 is from $\varepsilon_{2_{\min}}$ to $\varepsilon_{2_{\max}}$, ε_3 is from $\varepsilon_{3_{\min}}$ to $\varepsilon_{3_{\max}}$, R is from $R_{\min}$ to $R_{\max}$, and ω is from $\omega_{\min}$ to $\omega_{\max}$. The inequalities shown in (28) need to be met to ensure that the parameter constraints hold.

Inequalities Needed for the Parameter Constraints. Consider the following:

$$\begin{aligned} &v^2_{\max} t^2_{\max} + \varepsilon^2_{2_{\max}} + 2\varepsilon_{2_{\max}} v_{\max} \sin\alpha_{\max} t_{\max} \\ &\quad \le \varepsilon^2_{3_{\min}} + 2R_{\min}\left(\varepsilon_{3_{\min}} - \varepsilon_{2_{\max}} - v_{\max}\sin\alpha_{\max} t_{\max}\right) \\ &\left(R_{\max} - \varepsilon_{3_{\min}}\right)^2 \le \left(R_{\min} - \varepsilon_{2_{\max}} - v_{\max} t_{\max}\right)^2 \\ &\quad + 2\left(R_{\min} - \varepsilon_{2_{\max}}\right) v_{\min} t_{\min} \left(1 - \sin\alpha_{\max}\right) \\ &\omega_{\max} t_{\max} \le \alpha_{\min} \\ &\left(R_{\max} - \varepsilon_{1_{\min}}\right)^2 \le v^2_{\min} t^2_{\text{sample}} + \left(R_{\min} - \varepsilon_{2_{\max}}\right)^2 \\ &\quad + 2\left(R_{\min} - \varepsilon_{2_{\max}}\right) v_{\min} \sin\alpha_{\min} t_{\min} \\ &\left(R_{\min} + \varepsilon_{1_{\min}}\right)^2 \ge v^2_{\max} t^2_{\max} + \left(R_{\max} - \varepsilon_{2_{\min}}\right)^2 \\ &\quad + 2\left(R_{\max} - \varepsilon_{2_{\min}}\right) v_{\max} \sin\alpha_{\max} t_{\max}. \end{aligned} \tag{28}$$

It is obvious that the parameters $\varepsilon_{1_{\max}}, \varepsilon_{3_{\max}}$, and $\omega_{\min}$ do not appear in the constraint inequalities. Therefore, we increase ε_1 and ε_3 from the minimum and decrease ω from the maximum. We do not know the exact values of such parameters as $v, t_{\text{sample}}, \omega$, and can measure their values only by operating the vehicle. Errors cannot be avoided when we obtain these parameters. We can write the predicate logic formula asserting safety φ_{safety} as follows:

$$\begin{gathered} \forall\alpha \in \left[\alpha_{\min}, \alpha_{\max}\right], \\ \forall\varepsilon_1 \ge \varepsilon_{1_{\min}}, \quad \forall\varepsilon_2 \in \left[\varepsilon_{2_{\min}}, \varepsilon_{2_{\max}}\right], \quad \forall\varepsilon_3 \ge \varepsilon_{3_{\min}} \\ \forall R \in \left[R_{\min}, R_{\max}\right]: \\ \varphi_{\text{safety}} \wedge \varphi_1 \wedge \varphi_2 \wedge \varphi_3 \Longrightarrow \varphi^*_{\text{safety}}. \end{gathered} \tag{29}$$

Parameters v and ω can be viewed as the internal variables of the vehicle.

7. Conclusion

In this paper, we have modeled an AGV system using a hybrid I/O system and investigated a two-dimensional problem where the vehicle moves in a circular orbit. We derived and proved the constraints of the parameters of the AGV system so that the vehicle always move forward closely following the circular track and never moves backward. We have also analyzed the constraints of the parameters and the range of the parameters. Future research can extend this formulation from circular track to arbitrary complex curves, consider slopes or hilly terrains, and reason about multiple vehicle systems.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

The research for this study was supported by the NSFC (61373034, 61170304) and the International S&T Cooperation Program of China (2011DFG13000) (KZ201210028036).

References

[1] M. S. Branicky, "Introduction to hybrid systems," in *Handbook of Networked and Embedded Control Systems*, D. Hristu-Varsakelis and W. S. Levine, Eds., Control Engineering, pp. 91–116, Birkhäuser Boston, Boston, Mass, USA, 2005.

[2] A. Fehnker, F. Vaandrager, and M. Zhang, "Modeling and verifying a Lego car using hybrid I/O automata," in *Proceedings of the 3rd International Conference on Quality Software (QSIC '03)*, pp. 280–289, IEEE Computer Society, 2003.

[3] L. Balbis, A. W. Ordys, M. J. Grimble, and Y. Pang, "Tutorial introduction to the modelling and control of hybrid systems," *International Journal of Modelling, Identification and Control*, vol. 2, no. 4, pp. 259–272, 2007.

[4] N. Lynch, R. Segala, and F. Vaandrager, "Hybrid I/O automata," *Information and Computation*, vol. 185, no. 1, pp. 105–157, 2003.

[5] T. A. Henzinger, "Theory of hybrid automata," in *Proceedings of the 11th Annual IEEE Symposium on Logic in Computer Science (LICS '96)*, pp. 278–292, IEEE Computer Society, July 1996.

[6] J. Lygeros, G. Pappas, and S. Sastry, "An introduction to hybrid systems modeling, analysis and control," in *Proceedings of the 1st Nonlinear Control Network Pedagogical School*, pp. 307–329, Athens, Greece, 1999.

[7] A. Balluchi, L. Benvenuti, M. D. D. I. Benedetto, S. Member, C. Pinello, and A. Luigi, "Automotive engine control and hybrid systems: challenges and opportunities," *Proceedings of the IEEE*, vol. 88, no. 7, pp. 888–912, 2000.

[8] R. Alur, R. Grosu, Y. Hur, V. Kumar, and I. Lee, "Modular specification of hybrid systems in charon," in *Hybrid Systems: Computation and Control*, N. Lynch and B. Krogh, Eds., vol. 1790 of *Lecture Notes in Computer Science*, pp. 6–19, Springer, Berlin, Germany, 2000.

[9] M. Song, T. Tarn, and N. Xi, "Integration of task scheduling, action planning, and control in robotic manufacturing systems," *Proceedings of the IEEE*, vol. 88, no. 7, pp. 1097–1107, 2000.

[10] M. Katara, "Hybrid models for mobile computing," in *Coordination Languages and Models*, A. Porto and G. C. Roman, Eds., vol. 1906 of *Lecture Notes in Computer Science*, pp. 216–231, Springer, Berlin, Germany, 2000.

[11] B. Lennartson, M. Tittus, B. Egardt, and S. Pettersson, "Hybrid systems in process control," *IEEE Control Systems Magazine*, vol. 16, no. 5, pp. 45–56, 1996.

[12] N. A. Lynch and M. R. Tuttle, "An introduction to input/output automata," *CWI Quarterly*, vol. 2, no. 3, pp. 219–246, 1989.

[13] T. Muller, *Automated Guided Vehicles*, IFS, Kempston, UK, 1983.

[14] T. Le-Anh and M. B. M. de Koster, "A review of design and control of automated guided vehicle systems," *European Journal of Operational Research*, vol. 171, no. 1, pp. 1–23, 2006.

[15] I. F. A. Vis, "Survey of research in the design and control of automated guided vehicle systems," *European Journal of Operational Research*, vol. 170, no. 3, pp. 677–709, 2006.

[16] W. Kang, N. Xi, and J. Tan, "Analysis and design of non-time based motion controller for mobile robots," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA '99)*, pp. 2964–2969, May 1999.

[17] J. Tan, N. Xi, and W. Kang, "Non-time based tracking controller for mobile robots," in *Proceedings of the 1999 IEEE Canadian Conference on Electrical and Computer Engineering*, pp. 919–924, May 1999.

[18] R. Milner, *Communication and Concurrency*, Prentice-Hall, 1989.

[19] D. Park, "Concurrency and automata on infinite sequences," in *Theoretical Computer Science*, P. Deussen, Ed., vol. 104 of *Lecture Notes in Computer Science*, pp. 167–183, Springer, Berlin, Germany, 1981.

Hindawi Publishing Corporation
Journal of Applied Mathematics
Volume 2014, Article ID 512476, 9 pages
http://dx.doi.org/10.1155/2014/512476

Research Article

Modeling, Design, and Implementation of a Cloud Workflow Engine Based on Aneka

Jiantao Zhou,[1] Chaoxin Sun,[1] Weina Fu,[1] Jing Liu,[1] Lei Jia,[1] and Hongyan Tan[2]

[1] *College of Computer Science, Inner Mongolia University, Hohhot 010021, China*
[2] *High Performance Network Lab Institute of Acoustics, Chinese Academy of Sciences, Beijing 100190, China*

Correspondence should be addressed to Jiantao Zhou; zhoujiantao@tsinghua.org.cn

Received 28 January 2014; Accepted 11 March 2014; Published 29 April 2014

Academic Editor: X. Song

This paper presents a Petri net-based model for cloud workflow which plays a key role in industry. Three kinds of parallelisms in cloud workflow are characterized and modeled. Based on the analysis of the modeling, a cloud workflow engine is designed and implemented in Aneka cloud environment. The experimental results validate the effectiveness of our approach of modeling, design, and implementation of cloud workflow.

1. Introduction

With the successful cases of the world's leading companies, for example, Amazon and Google, cloud computing has become a hot topic in both industrial and academic areas. It embraces WEB 2.0, middleware, virtualization, and other technologies and also develops upon grid computing, distributed computing, parallel computing, and utility computing, and so forth [1]. Comparing with classic computing paradigms, cloud computing provides "a pool of abstracted, virtualized, dynamically scalable, managed computing power, storage, platforms, and services are delivered on demand to external customers over the Internet" [2]. It can provide scalable resources conveniently, on demand to different system requirements. According to features of services mainly delivered by the established cloud infrastructures, researchers separated the services into three levels, which are infrastructure as a service (IaaS), platform as a service (PaaS), and software as a service (SaaS). IaaS offers hardware resources and computing power, such as Amazon S3 for storage and EC2 for computing power. PaaS targets providing entire facilities including hardware and the application development environment, such as Microsoft Azure Services platform and Google App Engine. SaaS refers to those software applications offered as services in cloud environments.

However, along with the development of cloud computing, corresponding issues are also arising in both theoretical and technical aspects. One of the most prominent problems is how to minimize running costs and maximize revenues on the premise of maintaining or even improving the quality of service (QoS) [3]. Workflow technology can be regarded as one of the solutions [2–4]. A workflow is defined as "the automation of a business process, in whole or part, during which documents, information, or tasks are passed from one participant to another for action, according to a set of procedural rules" by the workflow management coalition (WFMC) [5]. Workflow management system (WFMS) is a system for defining, implementing, and managing the workflows, in which workflow engine is the most significant component for task scheduling, data movement, and exception handling.

Cloud workflow, also called cloud-based workflow [1] or cloud computing-oriented workflow [3], is a new application mode of workflow management system in the cloud environment, whose goal is to optimize the system performance, guarantee the QoS, and reduce running cost. It integrates workflow with cloud computing, which combined the advantages of both sides. Cloud workflow technology can be used in two levels [3]; from the cloud users' view, it supports process definitions of cloud applications and enables flexible configuration and automated operation of the processes. This kind of cloud workflow is regarded as "above-the-cloud," such as Microsoft Biztalk workflow service and IBM Lotuslive. From the cloud service providers' view, cloud workflow offers automatic task scheduling and resource management of cloud computing environment. This category is referred to as "in-the-cloud," for example, Cordys Process Factory (for Google APPs).

There is still much timely and worthwhile work to do in the field of cloud workflow, for example, scalability and load balance of above-the-cloud workflow, optimization and integration of in-the-cloud workflow, and so on. The goal of our work in this paper is to design an above-the-cloud workflow engine based on Aneka cloud platform [6], considering scalability and load balance. The remainder of the paper is organized as follows. Related work about workflow, grid workflow, and cloud workflow is given in Section 2. The core part is given in Section 3, including formal modeling of workflow processes and design of the Aneka-based cloud workflow engine. Then, the implementation and experiments are presented in Section 4. Effectiveness and correctness of the novel workflow engine are shown by analysis results. Finally, in Section 5, the main results of the paper are summarized.

2. Related Work

From the mid-1990s to around 2000, business process management and workflow technology were developed rapidly, and many valuable results and products were gained. The detailed surveys can be found in [7–9]. Besides, flexibility, adaptability, dynamic changes, and exception handling of workflow have also been focused on and taken progress in the next few years [10–13]. Thus, from the modeling and verification of workflow process, to the design and implementation of workflow engine, to the building and practice of workflow management system, a series of results have made workflow technology play a more and more important role in social life.

In the next ten years or so, with the emergence of new computer technologies and computing paradigms, for example, web services, P2P, and grid computing, workflow can be developed in two aspects. On the one hand, the existing workflows can be transferred to new computing paradigms. On the other hand, adapting to new features of technologies and paradigms, workflow technology should make a progress. Grid workflow can be classed into two categories, above-the-grid and in-the-grid, as the discussion of cloud workflow in the above section. The goal of in-the-grid workflow is integration, composition, and orchestration of grid services in the grid environment, considering peer-to-peer service interaction and complicated lifecycle management of grid services [14]. In regard to above-the-grid workflow, transformation of traditional business workflow systems is the part of the work, and many researches and practices have been done concerning distributed and parallel features of grid [15, 16], covering modeling, verification, scheduling problems, and so on [17–21].

Comparing cloud computing emerging in 2007 with grid computing [2], it can be seen that their visions are the same; meanwhile, there are both similarities and differences between them, from architecture, security model, business model, and computing model to provenance and applications. Based on the analysis of workflows running in these two kinds of infrastructure, similarities and differences are also found [3, 22]. There are many workflow studies on different levels of cloud services. Research of workflows building on IaaS focuses on dynamical deployment and monitoring in cloud nodes, which is used in large-scale data intensive computing [23, 24]. Some researchers use cloud workflows in community team working among multiple processes [25]. Otherwise, many workflow studies in PaaS pay attention to the integration of cloud and workflow. It is concerned with recognition and execution of workflow tasks in cloud environment [26, 27]. Today, there are also many studies on scientific and commercial cloud workflows. Yuan et al. studied data dependency and storage on scientific cloud workflows [28, 29]. Wu et al. carried out researches on hierarchical scheduling strategy in commercial cloud workflows [30].

According to our analysis, three kinds of differences or improvements can be concluded. Firstly, cloud workflow technology research is always carried out joint with multiple technologies and computing paradigms, such as web services, P2P, and grid computing. Secondly, cloud workflow concentrates more on data and resources and not just the control flow. Thirdly, performance of cloud workflow is paid more attention than functionalities [31–33].

3. Modeling and Design of Aneka-Based Cloud Workflow Engine

Our Aneka-based cloud workflow engine will be given in this section. To improve scalability, the cloud workflow engine will be designed to support different parallelism levels of workflow processes. And to clarify these parallelisms, the formal modeling technique for processes is given firstly.

3.1. Preliminaries. Workflow management system completely or partly supports automation of workflow schema. And workflow schema models business processes; it is characterized by the decomposition into subflows and atomic activities, the control flow between activities (subflows), data flow and data, and the assignment of resources (including human resources and equipment resources) to each activity [5]. That is, workflow schema is a combination of three essential dimensions: control flow, data flow, and resource flow.

Petri net is a simple, graphical, yet rigorous, mathematical formalism, which has been used to model workflow processes [34]. However, the usage of Petri net is always limited to model control flow, which is not enough for describing the above three dimensions of workflow. An advanced model founded on the basic Petri net has been developed in our previous paper [35], called 3DWFN. Its definition is given as follows, which is suitable for describing cloud workflow.

Definition 1. Three-Dimension WorkFlow Net, 3DWFN.

Let Σ, Γ, and Ψ be finite alphabet sets of activity names, data names, and resource names, respectively.

A three-dimension workflow net over them is a system $3DWFN = \langle S, T, F, C; Lab, Exp, M_0 \rangle$, where we have the following:

(i) finite sets S of places and T of *transitions*: $S \cap T = \emptyset$, $S \cup T \neq \emptyset$, and $T = Ta \cup Tp \cup T\tau$, where Ta is a set of *atomic transitions*, Tp is a set of *subnet transitions*, and $T\tau$ is a set of *internal transitions*;

(ii) $F \subseteq (S \times T) \cup (T \times S)$ is a set of arcs, especially, including the set of inhibitor arcs: $F^{\circ} \subseteq F$;

(iii) C is a finite and nonempty *color set* including types of tokens and the default is denoted by a black dot, "•;"

(iv) $Lab: T \rightarrow \Sigma, C \rightarrow \Gamma \cup \Psi \cup \Gamma \cup \Psi$ is a *labeling function*;

(v) $Exp: F \rightarrow N \times C \times Con$ is an *arc expression function*, where N denotes the set of natural numbers and Con is a set of expressions of numeric computation or logic computation;

(vi) $M_0 : \mathbf{S} \rightarrow \{\emptyset, \mu\mathbf{C}\}$ is the initial marking of the net, where $\mu\mathbf{C}$ is the multiset over C.

In view of defining transition rule of 3DWFN, some notations are introduced beforehand.

Definition 2. (i) *Projection*. Suppose $Fx : Q \rightarrow P_1 \times P_2 \times P_3 \times \cdots, \exists q \in Q, p_i \in P_i \ (i = 1, 2, 3, \ldots) : Fx(q) = (p_1, p_2, p_3, \ldots), Fx(q) \upharpoonright P_1 \times P_2$ represents $Fx(q)$'s projection on $P_1 \times P_2$, and $Fx(q) \upharpoonright P_1 \times P_2 = (p_1, p_2)$ or $\{p_1, p_2\}$.

(ii) *Variables Replacement*. Let "$x : A$" represent x is a variable of set A. Variable x may be replaced by any element in A.

Then the transition rule is given.

Definition 3. Transition Rule of 3DWFN.

(i) *Preconditions* about a transition t are denoted as for all $s \in {}^{\bullet}t, (s, t) \notin F^{\circ} : \mathrm{Pre}(s, t) = \mathrm{Exp}(s, t)$, $\mathrm{Pre}(t) = \bigcup_{s \in {}^{\bullet}t} \mathrm{Exp}(s, t)$, and for all $s \in {}^{\bullet}t, (s, t) \in F^{\circ} : \mathrm{Prev}(s, t) = \mathrm{Exp}(s, t)$.

(ii) *Postcondition* about a transition t is denoted as for all $s \in t^{\bullet}, : \mathrm{Post}(t, s) = \mathrm{Exp}(t, s), \mathrm{Post}(t) = \bigcup_{s \in t^{\bullet}} \mathrm{Exp}(t, s)$.

(iii) A transition $t \in T$ is *enabled* under a marking M if and only if (for all $s \in {}^{\bullet}t, (s, t) \notin F^{\circ} : M(s) \geq (\mathrm{Pre}(s, t) \upharpoonright \mathbf{N} \times \mathbf{C})) \wedge ($for all $s \in {}^{\bullet}t, (s, t) \in F^{\circ} : M(s) \prec (\mathrm{Prev}(s, t) \upharpoonright \mathbf{N} \times \mathbf{C}))$.

(iv) A new *marking* M' is produced after an enabled transition fires: $M' = M - (\mathrm{Pre}(t) \upharpoonright \mathbf{N} \times \mathbf{C}) + (\mathrm{Post}(t) \upharpoonright \mathbf{N} \times \mathbf{C})$ denoting as $M[t > M'$ or $M \xrightarrow{t} tM'$.

3.2. Modeling and Analysis of Workflow Process. With detailed analysis of generalized cloud workflow systems, it is really indispensable to discriminate different parallelisms of workflow processes for adopting cloud technologies to promote its execution efficiency. Therefore, we divide the execution of multiple tasks in a cloud workflow system into three levels according to different parallelisms, that is, process level execution, task level execution, and application level execution.

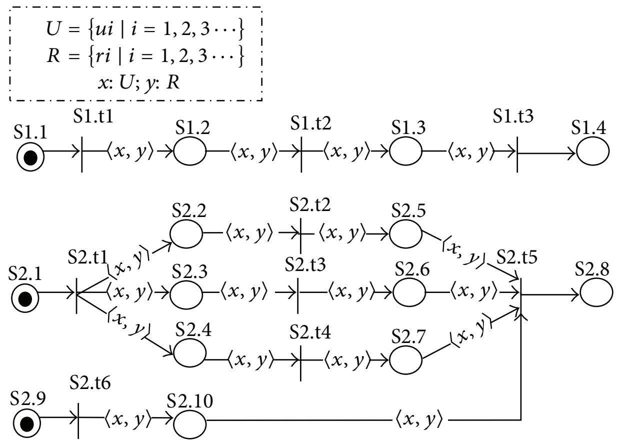


Figure 1: Parallel processes modeled by 3DWFN.

As shown in Figure 1 by 3DWFN, there are two parallel processes, named S1 and S2. S1 shows a sequential process and S2 shows a process including parallel tasks. U and R are colored sets. U represents the set of users, while R represents the set of resources. As a result, execution of a 3DWFN looks like a user acting, as some role carries some kind of data or uses some kind of resources to "walk" through a certain path of the net.

These two processes could be performed in parallel at different computing node in cloud environment, which illustrates the parallelism of process level execution in the cloud workflow system.

When the workflow system enter state S2.2, S2.3, and S2.4 after transition S2.t1 fired, three tasks that are represented by S2.t2, S2.t3, and S2.t4 are enabled to be carried out in parallel at different computing node in cloud environment controlled by a single user or multiple users. Then, the joint task S2.t5 could be triggered to be executed only if resources in S2.5, S2.6, S2.7, and S2.10 are all available. This scenario illustrates the parallelism of task level execution in the cloud workflow system.

Finally, if a single task is intensive computing, such as the task execution between S1.2 and S1.3, it could be divided into more fine-grained subtasks and carried out in parallel at different computing node in cloud environment controlled by a single user. This scenario shows the parallelism in application level.

3.3. Architecture of Cloud Workflow. According to the above-mentioned analysis, the integrated framework of Aneka-based cloud workflow engine is presented. There are three parts, environment, applications, and control parts, which can solve the analyzed parallelisms problems in the three levels and achieve extensibility and reusability of workflow. Details are presented in Figure 2.

3.3.1. Cloud Workflow Environment. There are three executing models in the Aneka cloud environment, Task Model, MapReduce Model, and Thread Model. In the remainder of this paper, all experiments are run with Task Model.

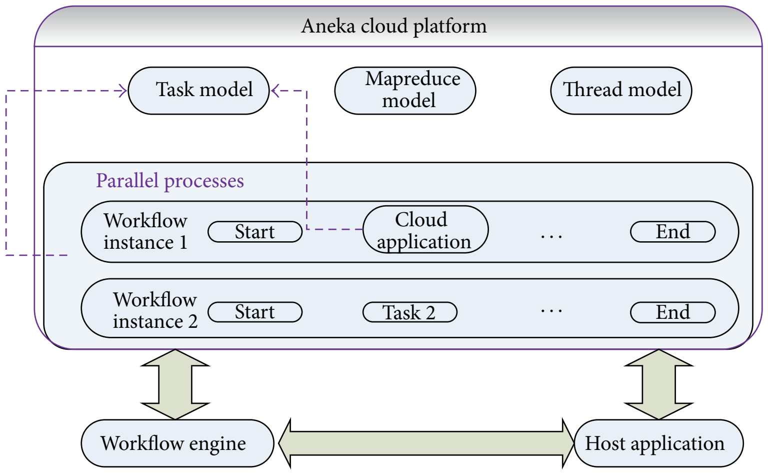


FIGURE 2: Integrated framework of Aneka-based cloud workflow engine.

3.3.2. Cloud Workflow Applications. The cloud workflow applications contain all instances of workflow processes.

3.3.3. Cloud Workflow Control. The cloud workflow control part contains the workflow engine and host application. The cloud workflow engine is in charge of running workflow instances, and the host application is in charge of defining workflow processes and monitoring workflow process executions.

3.4. Design of Cloud Workflow

3.4.1. Design of Workflow Runtime Environment. A lightweight cloud workflow engine is designed, whose functions include starting process execution, scheduling processes, and tasks based on preestablished rules. The workflow runtime environment is shown in Figure 3, which is composed of three parts, host application, workflow instances, and runtime engine.

Then, Figure 4 shows all service classes design of the workflow engine. The class "WorkflowRuntimeService" is the base class and the others are the derived classes.

3.4.2. Design of Cloud Workflow Execution Process. After a process is started, each task will be executed following the control flow when requirements of data flow and resource flow are met. When a task is enabled, the task executor will send its execution request to workflow engine. Then, the workflow engine will instantiate the ready task. If the task is not intensive computing, then it will be executed locally. Otherwise, if the task is intensive computing, it will be submitted to the cloud. The execution process of cloud workflow is depicted as in Figure 5.

3.4.3. Design of Task Model Submission Process. Next, the submission process of task is designed. As mentioned above, Task Model is chosen. In Aneka cloud environment, Task Model is used not only to solve the distributed applications which are composed of single tasks, but also to execute

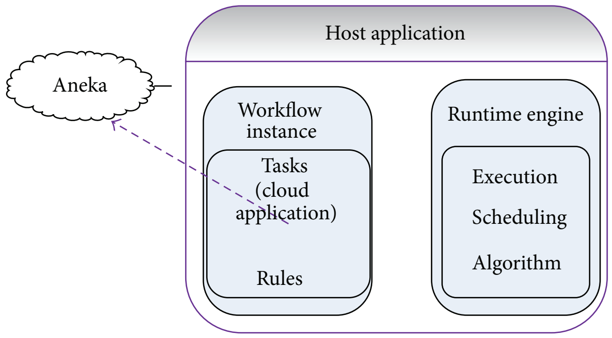


FIGURE 3: Workflow runtime environment.

the correlating tasks of these applications. Once users submit their sequence of tasks with rules, the results will be returned by Aneka after a while.

Aneka Task Model is composed of the class AnekaTask, the interface ITask, and the class AnekaApplication. The task submission process in Aneka Task Model is as follows: firstly, to define a class "UserTask," which inherits class "AnekaTask" in Aneka Task Model; secondly, to create an instance of UserTask for the application program; and thirdly, to package class "UserTask" instance to class "AnekaTask" and submit it to Aneka cloud by class "AnekaApplication." The sequence diagram of the above process is shown in Figure 6.

Then, the implementation of workflow tasks is presented in Figure 6. Firstly, the implementation manager submits tasks to the workflow runtime engine. Then, the workflow runtime engine selects a custom made operation flow and creates its workflow instance and then runs it at the same time.

4. Implementation and Experiment

4.1. Building Aneka Cloud Environment. Our experimental cloud environment includes a server as a master node, some common PCs as the worker nodes, and a manager

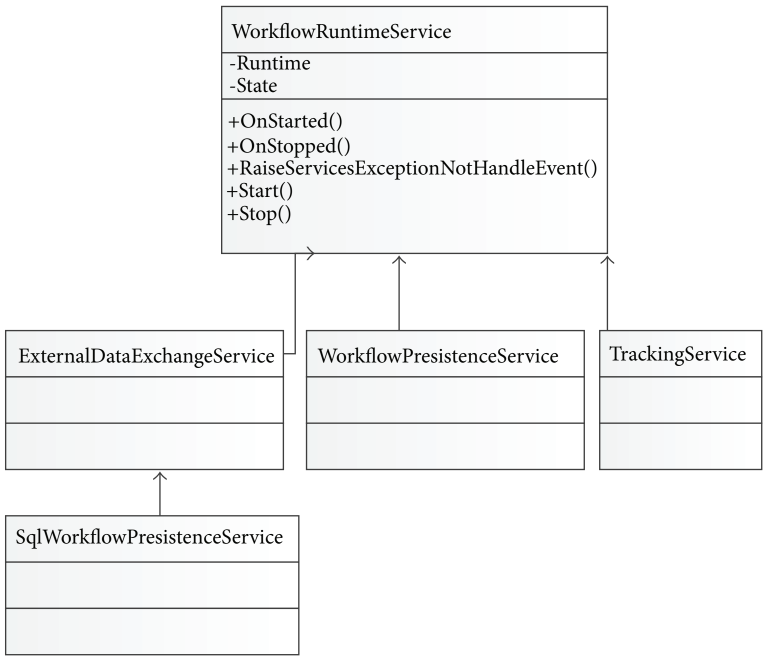

WorkflowRuntimeService
-Runtime
-State
+OnStarted()
+OnStopped()
+RaiseServicesExceptionNotHandleEvent()
+Start()
+Stop()
ExternalDataExchangeService
WorkflowPresistenceService
TrackingService
SqlWorkflowPresistenceService

FIGURE 4: Class diagram WF service.

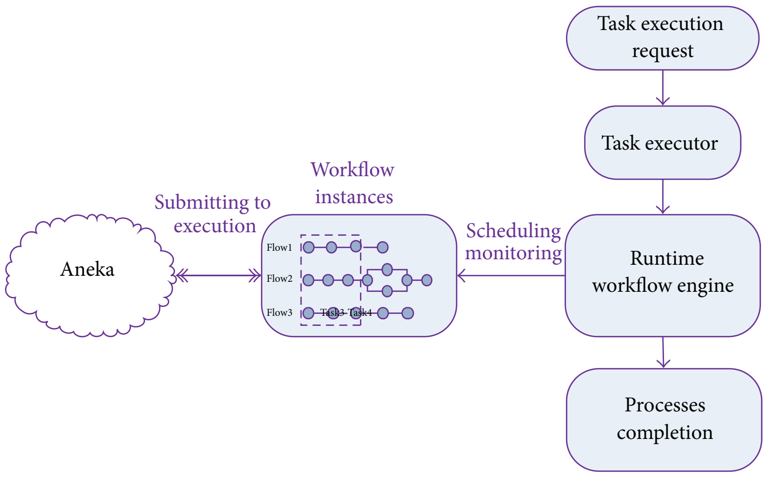


FIGURE 5: Process of cloud workflow execution.

node. In this paper, the detailed hardware configuration is presented in Table 1.

Then, the cloud nodes are pretreated according to the following steps before installing Aneka.

(a) Install FTP server on the master node and worker nodes.

(b) Offer access authority for the manager node to the master node and worker nodes.

After pretreatment, Aneka cloud management platform is installed in the manager node. Then, we use remote access to install and configure the master node and worker nodes.

TABLE 1: Aneka cloud environment configuration.

Type of node	Operating system	Quantity of computers
Manager	Windows 7	1
Master	Windows Server 2008	1
Worker	Windows 7	3

4.2. Workflow Process of an Example. Our example is to compute definite integral by a probabilistic method. The workflow process is composed of four steps: "generations of random number," "computing X-axis," "computing Y-axis," and "computation of final result."

In this processing, there are three correlating tasks, which are "generation of random numbers", "computation of X-axis x" and "computation of Y-axis y." We use the task "generation of random numbers" to compute real and

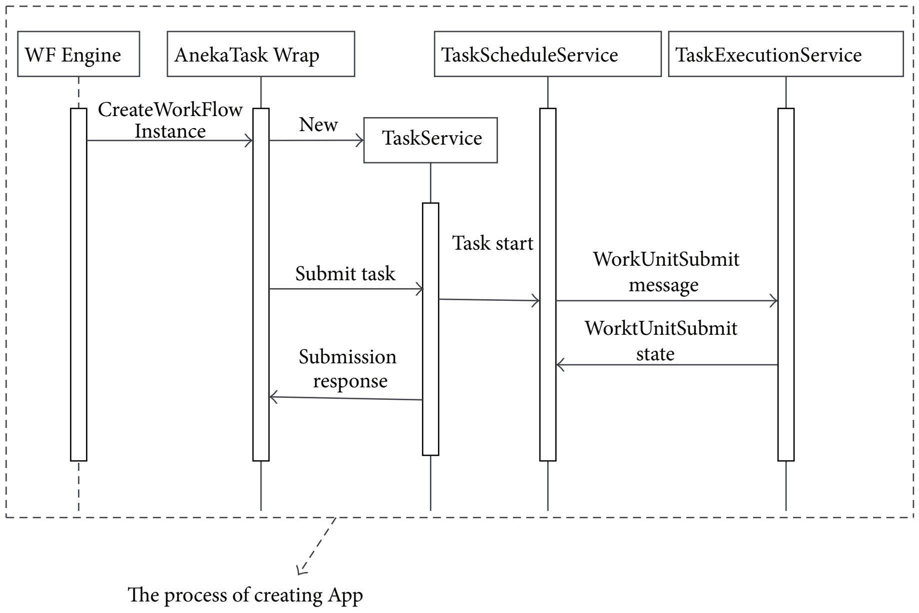


Figure 6: Process of submitting tasks.

imaginary axes. Then, we match and combine values of x, y to a *point*(x, y). Admittedly, the combinational step is next to the computational step of $f(x, y)$, and the steps are in one task. In this case, we divide the position relations of all points into two cases. When a $f(x, y)$ is between $x = a$ and $x = b$, we let the count increase by 1. In contrary, count is not changed. Then, letting the number of all points be equal to number n, we use $p = R/n$ to estimate $\int_a^b f(x)$. The operation flow is present in Figure 7.

4.3. Implementation Methods. The task of computing axes is intensive computing and will be submitted to the Aneka cloud in Task Model. The execution function under the interface ITask is presented in Algorithm 1. Then, the task is packaged and submitted to the cloud by the above mentioned class AnekaApplication. The core implementation is configured in Algorithm 2.

4.4. Results Analysis. Based on the above experiments, running results and workflow logs are analyzed. Three analysis conclusions are given as follows. Firstly, as shown in Table 2, the results of definite integral workflow are presented with different number of points. It shows that the degree of accuracy increases along with the increase of task's number. It accords with the mathematical regular rule. It is indicated that the functionality of our cloud workflow engine is normal.

Secondly, the intensive computing tasks submitted to the Aneka cloud are executed in parallel by different workers. As shown in Figure 8, the screenshot of workflow log, tasks A and B represent, respectively, the two intensive computing tasks, "computing X-axis" and "computing Y-axis." It is indicated that our cloud workflow engine is effective and efficient.

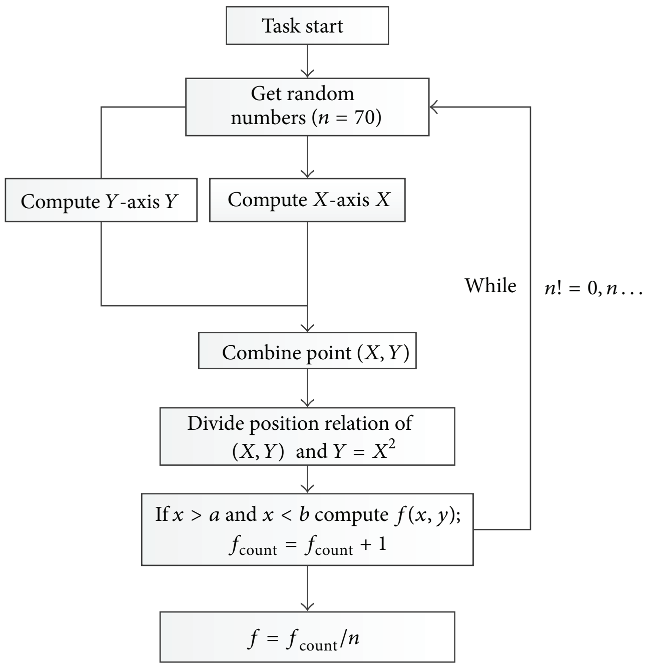

Task start
Get random numbers ($n = 70$)
Compute Y-axis Y
Compute X-axis X
While
$n! = 0, n\ldots$
Combine point (X, Y)
Divide position relation of (X, Y) and $Y = X^2$
If $x > a$ and $x < b$ compute $f(x, y)$; $f_{\text{count}} = f_{\text{count}} + 1$
$f = f_{\text{count}}/n$

Figure 7: Process diagram of task.

Table 2: Result of different number of the spots.

Number of the spots	70	100	150	1000
Computing result	0.286	0.310	0.393	0.342

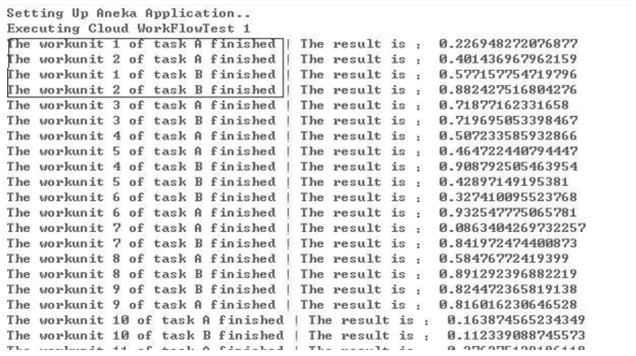
Thirdly, the running time of the whole workflow completion is not simple linear growth with the number of points used in definite integral. The screenshot of workflow log shown in Table 3 presents the time table of 10 tasks for their

```
public void Execute ()
{
this.result = (this.y * this.y) * this.rd;.
}
public void Execute ()
{
            this.result = this.x + (this.y – this.x) * this.rd:
}
```

Algorithm 1: Code of execute function.

```
private static AnekaApplication < AnekaTask, TaskManager > Setup (string [] args)
        {
            Configuration conf = Configuration.GetConfiguration (“configuration file”);
            // ensure that SingleSubmission is set to false
            // and that ResubmitMode to MANUAL.
            conf.SingleSubmission = false;
            conf.ResubmitMode = ResubmitMode.MANUAL;
            conf.UserCredential = new Aneka.Security.UserCredentials(“administrator”, “”);
            AnekaApplication < AnekaTask, TaskManager > app =
                  new AnekaApplication < AnekaTask, TaskManager > (“Workflow1”, conf);
            // ensure that SingleSubmission is set to false
            if (args.Length = = 1)
            {
                  bLogOnly = (args [0] = = “LogOnly”? true: false);
            }
            return app;
         }
|
```

Algorithm 2: Aneka application configuration.



```
Setting Up Aneka Application..
Executing Cloud WorkFlowTest 1
The workunit 1 of task A finished | The result is :  0.226948272076877
The workunit 2 of task A finished | The result is :  0.401436967962159
The workunit 1 of task B finished | The result is :  0.577157754719796
The workunit 2 of task B finished | The result is :  0.882427516804276
The workunit 3 of task A finished | The result is :  0.71877162331658
The workunit 3 of task B finished | The result is :  0.719695053398467
The workunit 4 of task A finished | The result is :  0.507233585932866
The workunit 5 of task A finished | The result is :  0.464722440794447
The workunit 4 of task B finished | The result is :  0.908792505463954
The workunit 5 of task B finished | The result is :  0.42897149195381
The workunit 6 of task B finished | The result is :  0.327410095523768
The workunit 6 of task A finished | The result is :  0.932547775065781
The workunit 7 of task A finished | The result is :  0.0863404269732257
The workunit 7 of task B finished | The result is :  0.841972474400873
The workunit 8 of task A finished | The result is :  0.58476772419399
The workunit 8 of task B finished | The result is :  0.891292396882219
The workunit 9 of task B finished | The result is :  0.824472365819138
The workunit 9 of task A finished | The result is :  0.816016230646528
The workunit 10 of task A finished | The result is :  0.163874565234349
The workunit 10 of task B finished | The result is :  0.112339088745573
```

Figure 8: Process of task execution.

execution time, waiting time, and total time. Mean execution time is about 2.6 seconds, and mean waiting time is about 1 second except the last task. The waiting time of the last task is so long that it is nearly equal to the sum time of the other 9 tasks. In this case, if the number of used points is increased in the next experiment, the total time might not be increased but decreased, because the cloud might assign more workers (resources) to execute them to save time. It is indicated that the cloud workflow engine is scalable, which will not spend an impossible large amount of time when a lot of tasks and processes run in parallel.

5. Conclusion

The work in this paper can be concluded as follows. A Petri net-based model called 3DWFN is given firstly, which can describe three dimensions of a workflow, that is, control flow, data flow, and resource flow. According to the analysis of the existing workflow systems, it is found that cloud workflow pays more attention to data, resources, and performance than control flow and functionality researched commonly in traditional workflow. Thus, 3DWFN is suitable for modeling of cloud workflow processes. Through analysis of the features of workflow processes executed potentially in cloud environment, three kinds of parallelisms are recognized as process level, task level, and application level and then modeled specifically by 3DWFN. The goal of the following design of cloud workflow engine is to support these parallelisms to improve scalability by using resources as far as in parallel.

Then, the architecture of the Aneka cloud based workflow engine is designed. The workflow runtime environment and execution process are stated, and the process of packaging and submitting an intensive computing task to Aneka is

Table 3: Timetable of task.

Total execution time	Total waiting time	Total time spent
00:00:02	00:00:00.5730000	00:00:02.5730000
00:00:06	00:00:01.2300000	00:00:07.2300000
00:00:02	00:00:00.6670000	00:00:02.6670000
00:00:02	00:00:01.3100000	00:00:03.3100000
00:00:03	00:00:00.8700000	00:00:03.8700000
00:00:04	00:00:00.0470000	00:00:04.0470000
00:00:03	00:00:00.4500000	00:00:03.4500000
00:00:01	00:00:02.9370000	00:00:03.9370000
00:00:01	00:00:00.7000000	00:00:01.7000000
00:00:02	00:06:019.5058631	00:06:15.9415449

explained. After the engine is implemented, a definite integral process is used as an example. By different numbers of points used to definite integral, the functionality and effectivity of the cloud workflow engine are shown. Based on the analysis of running results and workflow logs, the scalability and efficiency of the cloud workflow engine are given.

In the future, the research can be improved in following directions. First, more practical workflow processes will be designed using powerful expression of 3DWFN. Second, novel cloud workflow engine will be built, which has dynamic scheduling and handling functions with complicated process.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

This work is supported by Grants of the National Natural Science Foundation of China [nos. 61262082 and 61261019], Key Project of Chinese Ministry of Education [no. 212025], and Inner Mongolia Science Foundation for Distinguished Young Scholars [2012JQ03]. The authors wish to thank the anonymous reviewers for their helpful comments in reviewing this paper.

References

[1] M. A. Vouk, "Cloud computing—issues, research and implementations," *Journal of Computing and Information Technology*, vol. 16, no. 4, pp. 235–246, 2008.

[2] I. Foster, Y. Zhao, I. Raicu, and S. Lu, "Cloud computing and grid computing 360-degree compared," in *Proceedings of the Grid Computing Environments Workshop (GCE '08)*, pp. 1–10, November 2008.

[3] X. Z. Chai and J. Cao, "Cloud computing oriented workflow technology," *Journal of Chinese Computer Systems*, vol. 33, no. 1, pp. 90–95, 2012.

[4] R. Buyya, J. Broberg, and A. M. Goscinski, Eds., *Cloud Computing: Principles and Paradigms*, vol. 87, John Wiley & Sons, 2010.

[5] D. Hollingsworth, "Workflow management coalition the workflow reference model," Tech. Rep. 68, Workflow Management Coalition, Hampshire, UK, 1993.

[6] R. N. Calheiros, C. Vecchiola, D. Karunamoorthy, and R. Buyya, "The Aneka platform and QoS-driven resource provisioning for elastic applications on hybrid clouds," *Future Generation Computer Systems*, vol. 28, no. 6, pp. 861–870, 2012.

[7] W. M. P. Van Der Aalst, A. H. M. Ter Hofstede, and M. Weske, "Business process management: a survey," in *Business Process Management*, vol. 2678 of *Lecture Notes in Computer Science*, pp. 1–12, Springer, Berlin, Germany, 2003.

[8] R. Lu and S. Sadiq, "A survey of comparative business process modeling approaches," in *Business Information Systems*, pp. 82–94, Springer, Berlin, Germany, 2007.

[9] E. M. Bahsi, E. Ceyhan, and T. Kosar, "Conditional workflow management: a survey and analysis," *Scientific Programming*, vol. 15, no. 4, pp. 283–297, 2007.

[10] H. Schonenberg, R. Mans, N. Russell, N. Mulyar, and W. Van Der Aalst, "Process flexibility: a survey of contemporary approaches," in *Advances in Enterprise Engineering I*, Lecture Notes in Business Information Processing, pp. 16–30, Springer, Berlin, Germany, 2008.

[11] S. Smanchat, S. Ling, and M. Indrawan, "A survey on context-aware workflow adaptations," in *Proceedings of the 6th International Conference on Advances in Mobile Computing and Multimedia (MoMM '08)*, pp. 414–417, ACM, November 2008.

[12] S. Rinderle, M. Reichert, and P. Dadam, "Correctness criteria for dynamic changes in workflow systems—a survey," *Data and Knowledge Engineering*, vol. 50, no. 1, pp. 9–34, 2004.

[13] F. Casati, S. Ceri, S. Paraboschi, and G. Pozzi, "Specification and implementation of exceptions in workflow management systems," *ACM Transactions on Database Systems*, vol. 24, no. 3, pp. 405–451, 1999.

[14] S. Krishnan, P. Wagstrom, and G. Von Laszewski, "GSFL: a workflow framework for grid services," Tech. Rep. ANL/MCS-P980-0802, Argonne National Laboratory, 2002.

[15] J. Yu and R. Buyya, "A taxonomy of workflow management systems for Grid computing," *Journal of Grid Computing*, vol. 3, no. 3-4, pp. 171–200, 2005.

[16] G. C. Fox and D. Gannon, "Special issue: workflow in grid systems," *Concurrency Computation Practice and Experience*, vol. 18, no. 10, pp. 1009–1019, 2006.

[17] C. Pautasso and G. Alonso, "Parallel computing patterns for grid workflows," in *Proceedings of the Workshop on Workflows in Support of Large-Scale Science (WORKS '06)*, pp. 1–10, IEEE, June 2006.

[18] F. Lautenbacher and B. Bauer, "A survey on workflow annotation & composition approaches," in *Proceedings of the Workshop*

on Semantic Business Process and Product Lifecycle Management (SemBPM '07), 2007.

[19] M. Wieczorek, A. Hoheisel, and R. Prodan, "Taxonomies of the multi-criteria grid workflow scheduling problem," in *Grid Middleware and Services*, pp. 237–264, Springer, New York, NY, USA, 2008.

[20] J. Yu, R. Buyya, and K. Ramamohanarao, "Workflow scheduling algorithms for grid computing," *Studies in Computational Intelligence*, vol. 146, pp. 173–214, 2008.

[21] J. Chen and Y. Yang, "A taxonomy of grid workflow verification and validation," *Concurrency Computation Practice and Experience*, vol. 20, no. 4, pp. 347–360, 2008.

[22] E. Deelman, "Grids and clouds: making workflow applications work in heterogeneous distributed environments," *International Journal of High Performance Computing Applications*, vol. 24, no. 3, pp. 284–298, 2010.

[23] S. Pandey, D. Karunamoorthy, K. K. Gupta, and R. Buyya, "Megha workflow management system for application workflows," in *Proceedings of the IEEE Science & Engineering Graduate Research Expo*, 2009.

[24] Q. Chen, L. Wang, and Z. Shang, "MRGIS: a MapReduce-enabled high performance workflow system for GIS," in *Proceedings of the 4th IEEE International Conference on eScience (eScience '08)*, pp. 646–651, December 2008.

[25] W. Li, "A Community Cloud oriented workflow system framework and its Scheduling Strategy," in *Proceedings of the 2nd IEEE Symposium on Web Society (SWS '10)*, pp. 316–325, August 2010.

[26] X. Liu, J. Chen, and Y. Yang, *Temporal QOS Management in Scientific Cloud Workflow Systems*, Elsevier, 2012.

[27] S. Pandey, D. Karunamoorthy, and R. Buyya, "Workflow engine for clouds," in *Cloud Computing: Principles and Paradigms*, R. Buyya, J. Broberg, and A. Goscinski, Eds., pp. 321–344, John Wiley & Sons, New York, NY, USA, 2011.

[28] D. Yuan, Y. Yang, X. Liu, G. Zhang, and J. Chen, "A data dependency based strategy for intermediate data storage in scientific cloud workflow systems," *Concurrency and Computation: Practice and Experience*, vol. 24, no. 9, pp. 956–976, 2012.

[29] D. Yuan, Y. Yang, X. Liu, and J. Chen, "On-demand minimum cost benchmarking for intermediate dataset storage in scientific cloud workflow systems," *Journal of Parallel and Distributed Computing*, vol. 71, no. 2, pp. 316–332, 2011.

[30] Z. Wu, X. Liu, Z. Ni, D. Yuan, and Y. Yang, "A market-oriented hierarchical scheduling strategy in cloud workflow systems," *Journal of Supercomputing*, vol. 63, no. 1, pp. 256–293, 2013.

[31] G. Juve, E. Deelman, G. B. Berriman, B. P. Berman, and P. Maechling, "An evaluation of the cost and performance of scientific workflows on amazon EC_2," *Journal of Grid Computing*, vol. 10, no. 1, pp. 5–21, 2012.

[32] A. Verma and S. Kaushal, "Deadline and budget distribution based cost-time optimization workflow scheduling algorithm for cloud," in *Proceedings of the IJCA on International Conference on Recent Advances and Future Trends in Information Technology (iRAFIT '12)*, pp. 1–4, 2012.

[33] J. Behl, T. Distler, F. Heisig, R. Kapitza, and M. Schunter, "Providing fault-tolerant execution of web-service-based workflows within clouds," in *Proceedings of the 2nd International Workshop on Cloud Computing Platforms (CloudCP '12)*, article 7, April 2012.

[34] W. M. P. Van Der Aalst, "The application of Petri nets to workflow management," *Journal of Circuits, Systems and Computers*, vol. 8, no. 1, pp. 21–66, 1998.

[35] J. Zhou and X. Ye, "A flexible control strategy on workflow modeling and enacting," in *Proceedings of the 8th International Conference Advanced Communication Technology (ICACT '06)*, pp. 1712–1716, Republic of Korea, February 2006.

Hindawi Publishing Corporation
Journal of Applied Mathematics
Volume 2014, Article ID 564976, 8 pages
http://dx.doi.org/10.1155/2014/564976

Research Article

Optimal Spatial Matrix Filter Design for Array Signal Preprocessing

Haiyong Zhang,[1] Dong Han,[1,2] Kai Ji,[1] Zhong Ren,[1] Chi Xu,[1] Lijuan Zhu,[1] and Hongyan Tan[2]

[1] *Department of Communication Engineering, Dalian Naval Academy, Dalian 116018, China*
[2] *Institute of Acoustics, Chinese Academy of Sciences, Beijing 100190, China*

Correspondence should be addressed to Dong Han; small__hand@163.com

Received 12 February 2014; Revised 21 March 2014; Accepted 24 March 2014; Published 17 April 2014

Academic Editor: Guiming Luo

An efficient technique of designing spatial matrix filter for array signal preprocessing based on convex programming was proposed. Five methods were considered for designing the filter. In design method 1, we minimized the passband fidelity subject to the controlled overall stopband attenuation level. In design method 2, the objective function and the constraint in the design method 1 were reversed. In design method 3, the optimal matrix filter which has the general mean square error was considered. In design method 4, the left stopband and the right stopband were constrained with specific attenuation level each, and the minimized passband fidelity was received. In design method 5, the optimization objective function was the sum of the left stopband and the right stopband attenuation levels with the weighting factors 1 and γ, respectively, and the passband fidelity was the constraints. The optimal solution of the optimizations above was derived by the Lagrange multiplier theory. The relations between the optimal solutions were analyzed. The generalized singular value decomposition was introduced to simplify the optimal solution of design methods 1 and 2 and enhanced the efficiency of solving the Lagrange multipliers. By simulations, it could be found that the proposed method was effective for designing the spatial matrix filter.

1. Introduction

Spatial matrix filter can be used for array signal preprocessing in direction of arrival (DOA) estimation and matched field processing (MFP). The signals from the interested area are reserved and the interferences from other areas are restrained by preprocessing [1–3]. The estimation accuracy and the probability of resolution can be enhanced enormously. In addition, the use of spatial matrix filtering technology makes it possible to estimate bearings for more than N narrow-band sources using an N-element array [4]. In line array sonar signal processing, this method can be used to suppress the platform noise which depresses the capability of the sonar seriously and keeps it unresolved for a long time [5, 6].

The spatial matrix filter evolves from matrix filter which is more powerful for filtering short data records than the finite impulse response (FIR) digital filters [7–11]. In [8, 9], a semi-infinite optimization programming was constructed for filter design, which minimized the mean square error between the actual response and the desired response in the passband to ensure the stopband attenuation over continuously stopband frequency satisfied given specification. The matrix filter was used directly in DOA estimation of a linear array for improving the estimation accuracy. It is an approximation of the real spatial matrix filter in which the stopband attenuation over continuous directions satisfied the given specification. In [7], a convex programming is constructed by using least square or minimax criterion for matrix filter design; the responses over discrete frequency points are considered. The same criterion is used for designing spatial matrix filter with application to MFP in passive sonar [1]. Unfortunately, the optimal filter had not been given. As complex solving theory was needed, the filters designed by [1, 7] could not be used efficiently in DOA estimation either. In a recent paper [10, 11], the optimal matrix filters were obtained over continuous frequency point by using convex programming, and the optimal matrix filter for short data records processing was given. References [2, 3] put forward the concept of generalized spatial prefiltering, in which a second-order cone programming (SOCP) was constructed. The spatial matrix filter was used in DOA estimation [12] and MFP [2, 3]. A heavy computation load was required for a large number of sensors as the variable amount in the SOCP was at least the square number of the sensor amount. It was difficult to obtain the filter when the sensors increase to large number, and a long time was consumed.

In this paper, an effective technique of designing the spatial matrix filter based on convex programming for array signal preprocessing in DOA estimation was proposed. Five design methods were considered. All the optimal solutions were given directly by using the Lagrange multiplier theory. Theoretical analysis shows that the proposed methods are more efficient than the previous methods. Numerical results were presented to illustrate the effectiveness of the methods.

2. Optimal Spatial Matrix Filter Design

Consider a uniform linear array with N elements. Assume that there are D narrow band signals with the same frequency ω_0 incident onto the sensor array from directions $\boldsymbol{\theta} = [\theta_1, \theta_2, \ldots, \theta_D]$. The received signals are given by

$$\mathbf{x}(t) = \mathbf{A}(\boldsymbol{\theta})\mathbf{s}(t) + \mathbf{n}(t), \tag{1}$$

where $\mathbf{x}(t) = [x_1(t), \ldots, x_N(t)]^T$ and $\mathbf{s}(t) = [s_1(t), \ldots, s_N(t)]^T$ are the N-dimensional source signals; $\mathbf{n}(t) = [n_1(t), \ldots, n_N(t)]^T$ is the N-dimensional noise; $\mathbf{A}(\boldsymbol{\theta}) = [\mathbf{a}(\theta_1), \ldots, \mathbf{a}(\theta_i), \ldots, \mathbf{a}(\theta_D)]$ is the $N \times D$ steering matrix, where $\mathbf{a}(\theta_i) = [1, e^{-j\omega_0 \Delta \sin(\theta_i)/c}, \ldots, e^{-j\omega_0 (N-1)\Delta \sin(\theta_i)/c}]^T$ is the steering vector, $(\cdot)^T$ denotes the transpose of a matrix, and Δ is the distance of neighbor sensors.

Assume that the passband, left stopband, and right stopband steering matrices are $\mathbf{V}_P = [\mathbf{v}_{p1}, \ldots, \mathbf{v}_{pi}, \ldots, \mathbf{v}_{pP}]$, $\mathbf{v}_{pi} \in \Omega_P$, $1 \le i \le P$, $\mathbf{V}_{S_1} = [\mathbf{v}_{s1}, \ldots, \mathbf{v}_{sj}, \ldots, \mathbf{v}_{sS_1}]$, $\mathbf{v}_{sj} \in \Omega_{S_1}$, $1 \le j \le S_1$, and $\mathbf{V}_{S_2} = [\mathbf{v}_{t1}, \ldots, \mathbf{v}_{tk}, \ldots, \mathbf{v}_{tS_2}]$, $\mathbf{v}_{tk} \in \Omega_{S_2}$, $1 \le k \le S_2$, respectively, where $\mathbf{v}_{pi}$, $\mathbf{v}_{sj}$, and $\mathbf{v}_{tk}$ are the ith, jth, and kth steering vectors of passband, left stopband, and right stopband. The stopband steering matrix $\mathbf{V}_S = \mathbf{V}_{S_1} \cup \mathbf{V}_{S_2}$. $P > N$, $S_1 > N$, and $S_2 > N$ are the discrete numbers of the passband, left stopband, and right stopband regions, respectively, $\Omega_P = [\theta_{p1}, \theta_{p2}]$, $\Omega_{S_1} = [-90^\circ, \theta_{s1})$, $\Omega_{S_2} = (\theta_{s2}, 90^\circ]$, and $\Omega_S = \Omega_{S_1} \cup \Omega_{S_2}$ for $-90^\circ < \theta_{s1} \le \theta_{p1} < \theta_{p2} \le \theta_{s2} < 90^\circ$. $S = S_1 + S_2$ is the number of stopband steering vectors.

The filtering operation can be expressed as

$$\mathbf{z}(t) = \mathbf{H}\mathbf{x}(t) = \mathbf{H}\mathbf{A}(\boldsymbol{\theta})\mathbf{s}(t) + \mathbf{H}\mathbf{n}(t), \tag{2}$$

where $\mathbf{z}(t)$ are the N-dimensional output signals and $\mathbf{H}$ is an $N \times N$ spatial matrix filter. By preprocessing, the normalized passband fidelity and the normalized stopband attenuation level can be given by $\|\mathbf{H}\mathbf{V}_P - \mathbf{V}_P\|_F^2/NP$ and $\|\mathbf{H}\mathbf{V}_S\|_F^2/NS$, respectively. Similarly, the left stopband and the right stopband attenuation levels are $\|\mathbf{H}\mathbf{V}_{S_1}\|_F^2/NS_1$ and $\|\mathbf{H}\mathbf{V}_{S_2}\|_F^2/NS_2$, where $\|\cdot\|_F$ denotes the Frobenius norm of a matrix. The objective of a spatial matrix filter is to pass the interested signal over the passband and suppress the interference over the stopband. There are five methods to design the spatial matrix filter based on convex programming.

2.1. Design Method 1 (DM1). In this method, we minimize the normalized passband fidelity subject to the controlled normalized stopband attenuation constraint at the same time. In this optimization, the left stopband and the right stopband are treated as a whole:

$$\begin{aligned} \min_{\mathbf{H}_1} \quad & J(\mathbf{H}_1) = \frac{1}{NP}\|\mathbf{H}_1\mathbf{V}_P - \mathbf{V}_P\|_F^2 \\ \text{subject to} \quad & \frac{1}{NS}\|\mathbf{H}_1\mathbf{V}_S\|_F^2 \le \varepsilon_1, \end{aligned} \tag{3}$$

where ε_1 defines the stopband attenuation level. The average stopband attenuation is $10\log_{10}(\|\mathbf{H}_1\mathbf{V}_S\|_F^2/NS)$ decibels. When $\varepsilon_1 = 10^{k/10}$, then we will gain the average attenuation of k dB on the stopband. The average passband fidelity level and the left and the right stopband attenuation levels are chosen with the same way as follows in DM2, DM4, and DM5.

2.2. Design Method 2 (DM2). In this method, the objective function and the constraint in the design method 1 are reversed:

$$\begin{aligned} \min_{\mathbf{H}_2} \quad & J(\mathbf{H}_2) = \frac{1}{NS}\|\mathbf{H}_2\mathbf{V}_S\|_F^2 \\ \text{subject to} \quad & \frac{1}{NP}\|\mathbf{H}_2\mathbf{V}_P - \mathbf{V}_P\|_F^2 \le \xi_1, \end{aligned} \tag{4}$$

where ξ_1 defines the passband fidelity level.

2.3. Design Method 3 (DM3). The third method is to find out the optimal spatial matrix filter which has the general mean square error. In this method, the passband fidelity and stopband attenuation are treated exactly the same:

$$\min_{\mathbf{H}_3} J(\mathbf{H}_3) = \frac{\Delta\Omega_P}{NP}\|\mathbf{H}_3\mathbf{V}_P - \mathbf{V}_P\|_F^2 + \frac{\Delta\Omega_S}{NS}\|\mathbf{H}_3\mathbf{V}_S\|_F^2, \tag{5}$$

where $\Delta\Omega_P = \theta_{p2} - \theta_{p1}$ and $\Delta\Omega_S = 180^\circ + \theta_{s1} - \theta_{s2}$ are the widths of the passband and stopband regions, respectively.

2.4. Design Method 4 (DM4). The fourth method is to find the optimal matrix filter which constrains the left stopband and the right stopband with ε_2 and ε_3, respectively. When the interferences were differently on both sides, we could suppress them more precisely with different attenuation levels:

$$\begin{aligned} \min_{\mathbf{H}_4} \quad & J(\mathbf{H}_4) = \frac{1}{NP}\|\mathbf{H}_4\mathbf{V}_P - \mathbf{V}_P\|_F^2 \\ \text{subject to} \quad & \begin{cases} \dfrac{1}{NS_1}\left\|\mathbf{H}_4\mathbf{V}_{S_1}\right\|_F^2 \le \varepsilon_2 \\ \dfrac{1}{NS_2}\left\|\mathbf{H}_4\mathbf{V}_{S_2}\right\|_F^2 \le \varepsilon_3. \end{cases} \end{aligned} \tag{6}$$

2.5. Design Method 5 (DM5). Consider

$$\min_{\mathbf{H}_5} \quad J(\mathbf{H}_5) = \frac{1}{NS_1}\left\|\mathbf{H}_5\mathbf{V}_{S_1}\right\|_F^2 + \gamma\frac{1}{NS_2}\left\|\mathbf{H}_5\mathbf{V}_{S_2}\right\|_F^2$$
$$\text{subject to} \quad \frac{1}{NP}\left\|\mathbf{H}_5\mathbf{V}_P - \mathbf{V}_P\right\|_F^2 \le \xi_2. \tag{7}$$

In this method, we derive the sum of the left stopband and the right stopband attenuations with the weighting factors 1 and γ, respectively, and the passband fidelity is restricted by less than ξ_2. The weighting ratio of the right stopband and the left stopband attenuation was γ.

3. Problem Solution

In this section, the mathematical solutions of computing the optimal spatial matrix filters are derived by using the Lagrange multiplier technique. A numerical technique based on the generalized singular value decomposition method is also proposed for reducing the computational complexity of determining the optimal Lagrange multipliers.

For design method 1, the Lagrangian for the problem is defined as

$$L(\mathbf{H}_1;\lambda_1) = \frac{1}{NP}\left\|\mathbf{H}_1\mathbf{V}_P - \mathbf{V}_P\right\|_F^2 + \frac{\lambda_1}{NS}\left\|\mathbf{H}_1\mathbf{V}_S\right\|_F^2 - \lambda_1\varepsilon_1, \tag{8}$$

where $\lambda_1 > 0$ is the Lagrange multiplier. Because the cost function and the constraint are strictly convex, the Lagrangian is also convex in $\mathbf{H}_1$ and is minimized for any λ_1 by

$$\widehat{\mathbf{H}}_1(\lambda_1) = \mathbf{R}_P(\mathbf{R}_P + \lambda_1\mathbf{R}_S)^{-1}, \tag{9}$$

where $\mathbf{R}_P = \mathbf{V}_P\mathbf{V}_P^{\mathrm{H}}/NP$, $\mathbf{R}_S = \mathbf{V}_S\mathbf{V}_S^{\mathrm{H}}/NS$, $(\cdot)^{\mathrm{H}}$ denotes the conjugate transpose of a matrix, and $(\cdot)^{-1}$ denotes the inverse of a nonsingular matrix. Substitution of $\widehat{\mathbf{H}}_1(\lambda_1)$ into (9) gives the dual function

$$\phi(\lambda_1) \triangleq -\operatorname{Tr}\left[\mathbf{R}_P(\mathbf{R}_P + \lambda_1\mathbf{R}_S)^{-1}\mathbf{R}_P\right] + \operatorname{Tr}\left[\mathbf{R}_P\right] - \lambda_1\varepsilon_1, \tag{10}$$

where $\operatorname{Tr}(\cdot)$ denotes the trace of a matrix. The duality theorem states that the optimal Lagrange multiplier $\widehat{\lambda}_1$ can be obtained by maximizing $\phi(\lambda_1)$. It can be shown that the optimal spatial matrix filter and the equation of finding $\widehat{\lambda}_1$ are as

$$\widehat{\mathbf{H}}_1(\widehat{\lambda}_1) = \mathbf{R}_P\left(\mathbf{R}_P + \widehat{\lambda}_1\mathbf{R}_S\right)^{-1}, \tag{11}$$

$$\operatorname{Tr}\left[\mathbf{R}_P\left(\mathbf{R}_P + \widehat{\lambda}_1\mathbf{R}_S\right)^{-1}\mathbf{R}_S\left(\mathbf{R}_P + \widehat{\lambda}_1\mathbf{R}_S\right)^{-1}\mathbf{R}_P\right] = \varepsilon_1. \tag{12}$$

It is interesting to note from (12) that as $\varepsilon_1 \to 0$, $\widehat{\lambda}_1 \to \infty$. Hence, for a smaller value of ε_1, we obtain a larger Lagrange multiplier and vice versa. From (8), a larger value of $\widehat{\lambda}_1$ implies that the stopband error weighted more heavily compared to the passband error. Hence, one would expect that a larger value of $\widehat{\lambda}_1$ would lead to better stopband attenuation at the cost of increased passband ripple.

Similarly, for design method 2, the optimal spatial matrix filter and the equation of finding the optimal Lagrange multiplier $\widehat{\lambda}_2$ are as follows:

$$\widehat{\mathbf{H}}_2\left(\widehat{\lambda}_2\right) = \widehat{\lambda}_2\mathbf{R}_P\left(\widehat{\lambda}_2\mathbf{R}_P + \mathbf{R}_S\right)^{-1}, \tag{13}$$

$$1 - \operatorname{Tr}\left[\widehat{\lambda}_2\mathbf{R}_P\left(\widehat{\lambda}_2\mathbf{R}_P + \mathbf{R}_S\right)^{-1}\mathbf{R}_S\left(\widehat{\lambda}_2\mathbf{R}_P + \mathbf{R}_S\right)^{-1}\mathbf{R}_P\right]$$
$$- \operatorname{Tr}\left[\widehat{\lambda}_2\mathbf{R}_P\left(\widehat{\lambda}_2\mathbf{R}_P + \mathbf{R}_S\right)^{-1}\mathbf{R}_P\right] = \xi_1. \tag{14}$$

For design method 3, the optimal spatial matrix filter with the general mean square error can be deduced by (11) with $\widehat{\lambda}_1 = \Delta\Omega_S/\Delta\Omega_P$ or by (13) with $\widehat{\lambda}_2 = \Delta\Omega_P/\Delta\Omega_S$. The optimal spatial matrix filter is

$$\widehat{\mathbf{H}}_3 = \Delta\Omega_P\mathbf{R}_P(\Delta\Omega_P\mathbf{R}_P + \Delta\Omega_S\mathbf{R}_S)^{-1}. \tag{15}$$

Note that (13) can be reexpressed as

$$\widehat{\mathbf{H}}_2\left(\widehat{\lambda}_2\right) = \mathbf{R}_P\left(\mathbf{R}_P + \frac{1}{\widehat{\lambda}_2}\mathbf{R}_S\right)^{-1}. \tag{16}$$

Comparing (11) with (16), it is clear from (11) and (13) that if $1/\widehat{\lambda}_2 = \widehat{\lambda}_1$, then the two optimal spatial matrix filters are identical. When the stopband attenuation has been specified with ε by (12) with a certain $\widehat{\lambda}_1$, the passband fidelity ξ can be easily deduced by (14) with $\widehat{\lambda}_2 = 1/\widehat{\lambda}_1$ and vice versa. By using $\widehat{\lambda}_2 \times \widehat{\lambda}_1 = 1$, (12), and (14), the relationship between the stopband attenuation and passband fidelity is

$$1 = \xi_1 + \widehat{\lambda}_1\varepsilon_1 + \operatorname{Tr}\left[\mathbf{R}_P\left(\mathbf{R}_P + \widehat{\lambda}_1\mathbf{R}_S\right)^{-1}\mathbf{R}_P\right]. \tag{17}$$

The optimal spatial matrix filter with general mean square error is obtained by design method 3. By substituting $\widehat{\lambda} = \Delta\Omega_S/\Delta\Omega_P$ into (17), the general response error η is

$$\eta = \Delta\Omega_P\xi_1 + \Delta\Omega_S\varepsilon_1$$
$$= \Delta\Omega_P - \operatorname{Tr}\left[\Delta\Omega_P\mathbf{R}_P(\Delta\Omega_P\mathbf{R}_P + \Delta\Omega_S\mathbf{R}_S)^{-1}\Delta\Omega_P\mathbf{R}_P\right]. \tag{18}$$

It can be seen that this filter is just the same as the filter which is proposed by MacInnes in [4]. In paper [4], the filter was given by the product of one matrix and the pseudoinverse of another matrix.

Similarly, the optimal solution of design methods 4 and 5 can be derived by using the Lagrange theory. The optimal solution $\widehat{\mathbf{H}}_4$ and the equations for solving the optimal Lagrange multipliers $\widehat{\lambda}_1$ and $\widehat{\lambda}_2$ of design method 4 are given directly by the following:

$$\widehat{\mathbf{H}}_4 = \mathbf{R}_P\left(\mathbf{R}_P + \widehat{\lambda}_3\mathbf{R}_{S_1} + \widehat{\lambda}_4\mathbf{R}_{S_2}\right)^{-1} \tag{19}$$

$$\operatorname{Tr}\left[\mathbf{R}_P\left(\mathbf{R}_P + \widehat{\lambda}_3\mathbf{R}_{S_1} + \widehat{\lambda}_4\mathbf{R}_{S_2}\right)^{-1}\mathbf{R}_{S_1}\right.$$
$$\left.\cdot\left(\mathbf{R}_P + \widehat{\lambda}_3\mathbf{R}_{S_1} + \widehat{\lambda}_4\mathbf{R}_{S_2}\right)^{-1}\mathbf{R}_P\right] = \varepsilon_2 \tag{20}$$

$$\operatorname{Tr}\left[\mathbf{R}_P\left(\mathbf{R}_P + \widehat{\lambda}_3\mathbf{R}_{S_1} + \widehat{\lambda}_4\mathbf{R}_{S_2}\right)^{-1}\mathbf{R}_{S_2}\right.$$
$$\left.\cdot\left(\mathbf{R}_P + \widehat{\lambda}_3\mathbf{R}_{S_1} + \widehat{\lambda}_4\mathbf{R}_{S_2}\right)^{-1}\mathbf{R}_P\right] = \varepsilon_3. \tag{21}$$

The optimal solution of design method 5 and the equation for solving the optimal Lagrange multiplier are given as follows:

$$\widehat{\mathbf{H}}_5 = \widehat{\lambda}_5 \mathbf{R}_P \left(\widehat{\lambda}_5 \mathbf{R}_P + \mathbf{R}_{S_1} + \gamma \mathbf{R}_{S_2}\right)^{-1} \quad (22)$$

$$\begin{aligned}\operatorname{Tr}\Big[&\widehat{\lambda}_5 \mathbf{R}_P \left(\widehat{\lambda}_5 \mathbf{R}_P + \mathbf{R}_{S_1} + \gamma \mathbf{R}_{S_2}\right)^{-1} \\ &\times \mathbf{R}_{S_2} \left(\widehat{\lambda}_5 \mathbf{R}_P + \mathbf{R}_{S_1} + \gamma \mathbf{R}_{S_2}\right)^{-1} \widehat{\lambda}_5 \mathbf{R}_P\Big] \\ &= -\operatorname{Tr}(\mathbf{R}_P) + \xi_2 \\ &\quad + 2\operatorname{Tr}\left[\widehat{\lambda}_5 \mathbf{R}_P \left(\widehat{\lambda}_5 \mathbf{R}_P + \mathbf{R}_{S_1} + \gamma \mathbf{R}_{S_2}\right)^{-1} \mathbf{R}_P\right].\end{aligned} \quad (23)$$

For a given division of the passband and the stopband, the correlation matrices $\mathbf{R}_P$, $\mathbf{R}_S$, $\mathbf{R}_{S_1}$, and $\mathbf{R}_{S_2}$ could be obtained. If we get the optimal Lagrange multiplier, then we get the optimal spatial matrix filters by (11), (13), (19), and (22). In other words, the spatial matrix filter design problems had been changed into the problems of solving the nonlinear equations of (12), (14), (20), (21), and (23). There were only 1 (DM1, DM2, and DM5) or 2 (DM4) parameters that need to be solved. The optimizations of the previous methods besides [4] (the optimal filter was the same as that of DM3) were very complicated. The unknown $N \times N$ filter $\mathbf{H}$ has N^2 unknown parameters that need to be solved. In addition, with the increasing of the discrete points in the passband and the stopband region, the computational complexity of other methods increased greatly, whereas it had little effect on the proposed methods. The proposed methods were very efficient for spatial matrix filter design.

Noticing that all the equations for solving the multipliers were monotonous nonlinear functions, for the given stopband attenuation level and the passband fidelity level, the Lagrange multipliers were unique. They could be obtained by the most iterative root finding algorithm, whereas the other methods need some professional optimization software. In this paper, the method of dichotomy was used.

4. Improving the Design Efficiency of Design Methods 1 and 2

The efficiency of designing the optimal matrix filter in design method 1 or in design method 2 is mainly influenced by determining the optimal Lagrange multiplier $\widehat{\lambda}$ or $\widehat{\lambda}'$. The computational complexity can be reduced significantly by using the generalized singular value decompensation method [13–15]. Since $\mathbf{V}_P$ and $\mathbf{V}_S$ are nonsingular Vandermonde matrices, there exist unitary matrices $\mathbf{U}_P \in \mathbf{C}^{P\times P}$ and $\mathbf{U}_S \in \mathbf{C}^{S\times S}$ and a nonsingular matrix $\mathbf{Q}_X \in \mathbf{C}^{N\times N}$ such that

$$\begin{aligned}\mathbf{V}_P &= \mathbf{Q}_X^{-\mathrm{H}} \left[\boldsymbol{\Sigma}_P, \mathbf{0}_{N\times(P-N)}\right] \mathbf{U}_P \\ \mathbf{V}_S &= \mathbf{Q}_X^{-\mathrm{H}} \left[\boldsymbol{\Sigma}_S, \mathbf{0}_{N\times(S-N)}\right] \mathbf{U}_S,\end{aligned} \quad (24)$$

where $\boldsymbol{\Sigma}_P = \operatorname{diag}(\alpha_1, \ldots, \alpha_i, \ldots, \alpha_N)$, $\boldsymbol{\Sigma}_S = \operatorname{diag}(\beta_1, \ldots, \beta_i, \ldots, \beta_N)$, and $\alpha_i^2 + \beta_i^2 = 1$, $i = 1, 2, \ldots, S$.

By substitution of (24) into (11), (13), and (15), the optimal spatial matrix filters can be given by

$$\begin{aligned}\widehat{\mathbf{H}}_1 &= \mathbf{Q}_X^{-\mathrm{H}} \frac{\boldsymbol{\Sigma}_P^2}{NP}\left(\frac{\boldsymbol{\Sigma}_P^2}{NP} + \widehat{\lambda}\frac{\boldsymbol{\Sigma}_S^2}{NS}\right)^{-1} \mathbf{Q}_X^{\mathrm{H}} \\ \widehat{\mathbf{H}}_2 &= \widehat{\lambda}_2 \mathbf{Q}_X^{-\mathrm{H}} \frac{\boldsymbol{\Sigma}_P^2}{NP}\left(\widehat{\lambda}_2\frac{\boldsymbol{\Sigma}_P^2}{NP} + \frac{\boldsymbol{\Sigma}_S^2}{NS}\right)^{-1} \mathbf{Q}_X^{\mathrm{H}} \\ \widehat{\mathbf{H}}_3 &= \mathbf{Q}_X^{-\mathrm{H}} \frac{\boldsymbol{\Sigma}_P^2}{NP}\left(\frac{\boldsymbol{\Sigma}_P^2}{NP} + \frac{\boldsymbol{\Sigma}_S^2}{NS}\right)^{-1} \mathbf{Q}_X^{\mathrm{H}}.\end{aligned} \quad (25)$$

For design methods 1 and 2, the optimal Lagrange multipliers $\widehat{\lambda}_1$ and $\widehat{\lambda}_2$ are the roots of the following equations, respectively:

$$\operatorname{Tr}\left[\mathbf{Q}_X^{-\mathrm{H}} \frac{\boldsymbol{\Sigma}_P^4 \boldsymbol{\Sigma}_S^2}{(NP)^2 NS}\left(\frac{\boldsymbol{\Sigma}_P^2}{NP} + \widehat{\lambda}_1\frac{\boldsymbol{\Sigma}_S^2}{NS}\right)^{-2} \mathbf{Q}_X^{-1}\right] = \varepsilon_1 \quad (26)$$

$$\begin{aligned}1 - \operatorname{Tr}&\left[\widehat{\lambda}_2 \mathbf{Q}_X^{-\mathrm{H}} \frac{\boldsymbol{\Sigma}_P^4 \boldsymbol{\Sigma}_S^2}{(NP)^2 NS}\left(\widehat{\lambda}_2\frac{\boldsymbol{\Sigma}_P^2}{NP} + \frac{\boldsymbol{\Sigma}_S^2}{NS}\right)^{-2} \mathbf{Q}_X^{-1}\right] \\ - \operatorname{Tr}&\left[\widehat{\lambda}_2 \mathbf{Q}_X^{-\mathrm{H}} \frac{\boldsymbol{\Sigma}_P^4}{(NP)^2}\left(\widehat{\lambda}_2\frac{\boldsymbol{\Sigma}_P^2}{NP} + \frac{\boldsymbol{\Sigma}_S^2}{NS}\right)^{-1} \mathbf{Q}_X^{-1}\right] = \xi_1.\end{aligned} \quad (27)$$

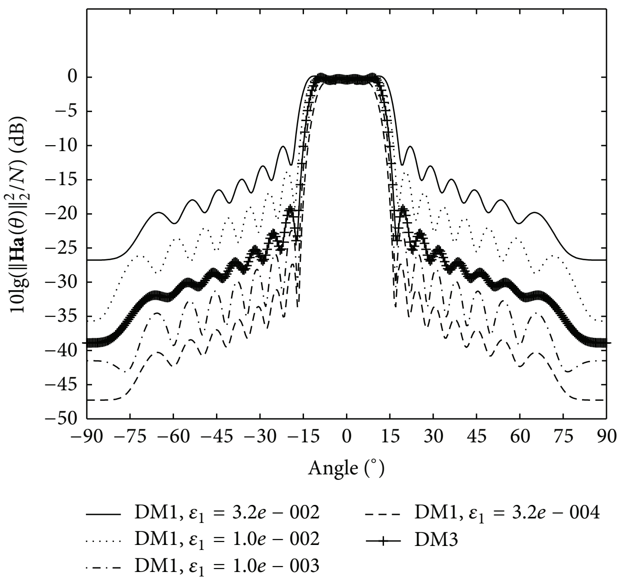
Any root finding method can be used to solve for $\widehat{\lambda}_1$ and $\widehat{\lambda}_2$ which satisfy the equations defined by (26) and (27), respectively. Note that if (12) or (14) is solved directly using some iterative methods, it requires a matrix inversion $(\mathbf{R}_P + \widehat{\lambda}_1 \mathbf{R}_S)^{-1}$ or $(\widehat{\lambda}_2 \mathbf{R}_P + \mathbf{R}_S)^{-1}$ for each iteration. It is of the order of N^3 computations. On the other hand, using (26) or (27), the computation load becomes trivial because the matrix inversion involves a diagonal matrix. The generalized singular value decomposition of two matrices is the only price needed to be paid for the computation load. However, this needs to be done only once. Therefore, the computation efficiency can be greatly enhanced.

5. Computer Simulation

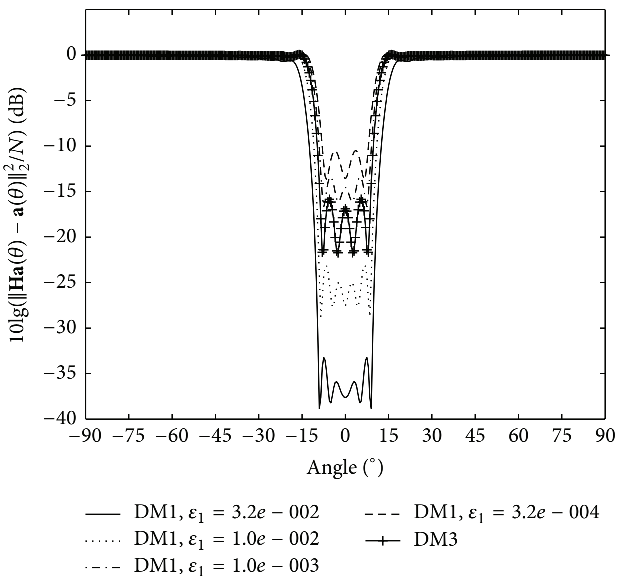
In this section, the passband and stopband of all the filters are specified with $\Omega_P = [-10^\circ, 10^\circ]$ and $\Omega_S = \Omega_{S_1} \cup \Omega_{S_2} = [-90^\circ, -15^\circ] \cup [15^\circ, 90^\circ]$. The filters have the same dimension with $N = 20$.

Figure 1 showed the filters design by using DM1 and DM3. The stopband fidelities ε_1 of DM1 filters were 0.032, 0.01, 0.001, and 0.00032, respectively, which were −15 dB, −20 dB, −30 dB, and −35 dB in decibels. The passband fidelity and stopband attenuation level of the filter designed by DM3 were 0.0222 and 0.0026, respectively, which were −16.53 dB and −25.83 dB in decibels. As we know, filter of DM3 has the general mean square response error, and it was obvious that the filter of DM3 can be obtained by using DM1 with properly set stopband attenuation level.

In paper by Zhu et al. [8], a semi-infinite optimization programming was used to obtain the optimal matrix filter.

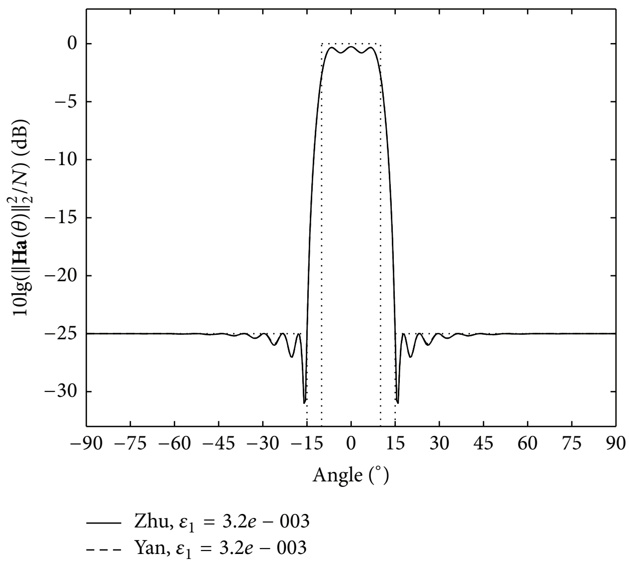


(a) Power response

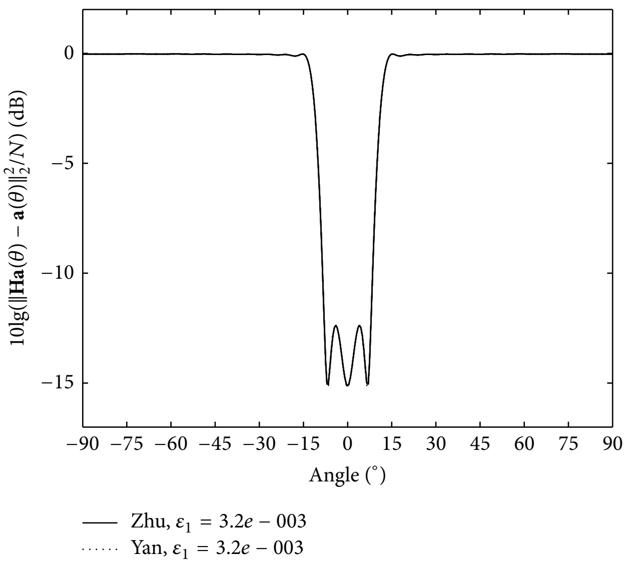


(b) Mean square error

FIGURE 1: Characteristics of the spatial matrix filters designed by DM1 and DM3.



(a) Power response



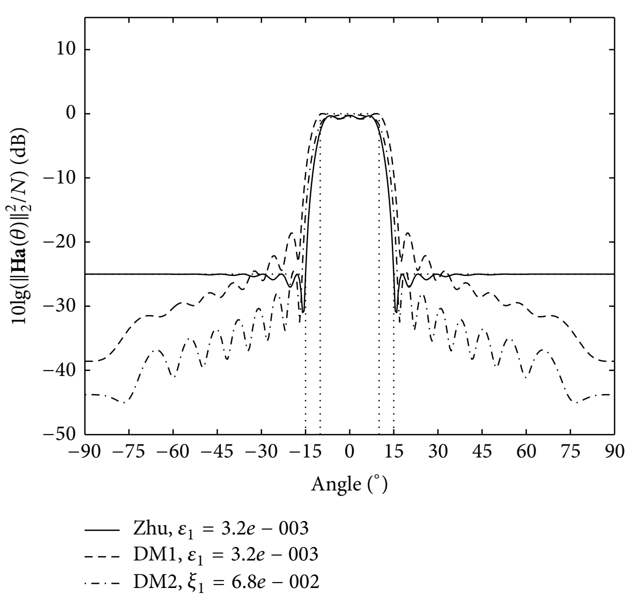
(b) Mean square error

FIGURE 2: The comparison of Zhu method [8] with Yan method [3].

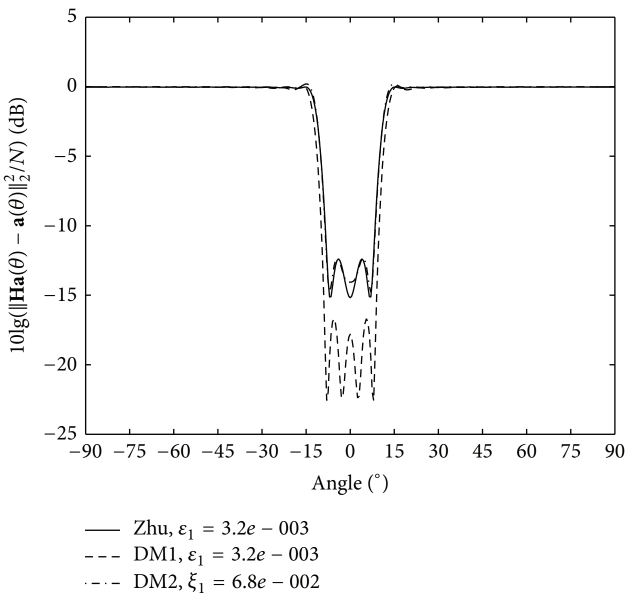
Although the filter was designed for digital filtering, it was used directly for spatial filtering in direction of arrival estimation. Yan and Ma [3] designed the specific spatial matrix filter for array signal preprocessing in direction of arrival estimation (Figure 2). Figure 3 compared the two filters designed above. As we could see there was a negligible difference between the two filters. And we need only to compare the Zhu filter with the proposed filter to illustrate the effectiveness of the proposed method.

Figure 3 showed the filters designed with Zhu method [8], DM1, and DM2. The filters by Zhu method and DM1 had the same average attenuation level with −25 dB in the stopband. In DM2, the passband fidelity was $\xi_1 = 0.06823$ (−11.66 dB), which was the same as that of the filter obtained by using Zhu method. It could be seen from Figure 3, on the one hand, that the filter of DM1 has much less passband response error than that of Zhu method. On the other hand, the filter of DM2 has much lower stopband attenuation than that of Zhu method.

As we had discussed above, by a smaller value of ε_1, we obtain a larger Lagrange multiplier λ_1 and vice versa. Similarly, by a smaller value of ξ_1, we could also obtain a larger Lagrange multiplier λ_2 and vice versa. Figure 4 gave



(a) Power response



(b) Mean square error

Figure 3: The comparison of Zhu method [8] with DM1 and DM2.

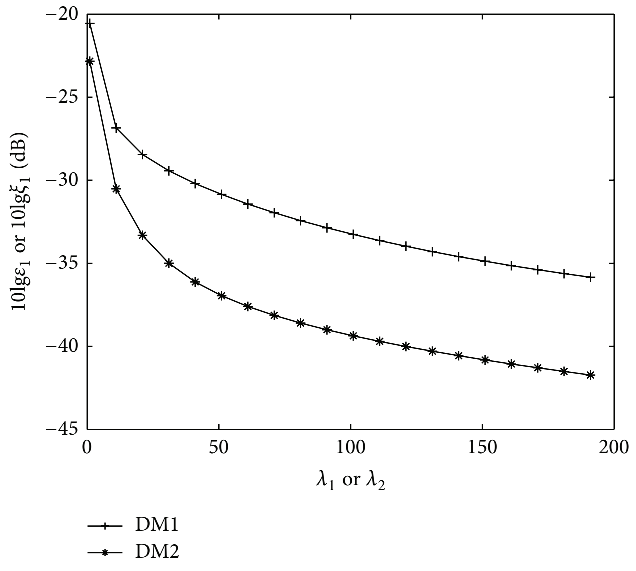


Figure 4: The relationship between the optimal Lagrange multiplier λ_1 (λ_2) and the stopband attenuation ε_1 (the passband fidelity ξ_1), in which the filters were designed by DM1 (DM2).

the relationship between the optimal Lagrange multiplier and the passband fidelity or the stopband attenuation, in which the filters were designed by using DM1 and DM2. It was obvious that the relation models between them were nonlinear monotonous functions. The optimal Lagrange multiplier was uniquely for the given passband fidelity or stopband attenuation. The multipliers could be obtained by the most iterative root finding algorithm. The dichotomy method was used here for solving the equations.

Figure 5 gave the characteristics of the spatial matrix filters designed by DM4. The left stopband attenuation level ε_2 was 0.01, −20 dB in decibels. The right stopband attenuation levels were 0.032, 0.01, and 0.0032, respectively, which were −15 dB, −20 dB, and −25 dB in decibels, respectively. Compared with the first three methods, design method 4 could restrain the left stopband and the right stopband separately, while the methods of design methods 1 to 3 could only treat the stopband as a whole. The filters of DM4 were more effective when the interferences we need to suppress had different signal-to-noise ratio in the left and right stopbands.

The relationships between the optimal Lagrange multipliers $\widehat{\lambda}_1$, $\widehat{\lambda}_2$, $\widehat{\lambda}_3$, $\widehat{\lambda}_4$, and $\widehat{\lambda}_5$ and the stopband attenuation or the passband fidelity ε_1, ξ_1, ε_2, ε_3, and ξ_2 correspondingly were monotonic nonlinear. Figure 6 gave the relationship between the optimal Lagrange multiplier $\widehat{\lambda}_5$ and the passband fidelity ξ_2, in which the spatial matrix filters were designed by DM5. The right stopband attenuation weighting factors were $\gamma = 0.25$, $\gamma = 0.5$, $\gamma = 1$, $\gamma = 2$, and $\gamma = 4$. It could be seen from Figure 6 that one could obtain better passband fidelity by increasing the Lagrange multiplier.

6. Conclusion

An efficient technique of designing spatial matrix filter based on convex programming for array signal preprocessing was proposed. Five methods were considered. The mathematical solutions of computing the optimal spatial matrix filters were derived by using the Lagrange multiplier technique. A numerical technique based on the generalized singular value decomposition method was also proposed for reducing the computational complexity of determining the optimal Lagrange multipliers of the first two design methods. By

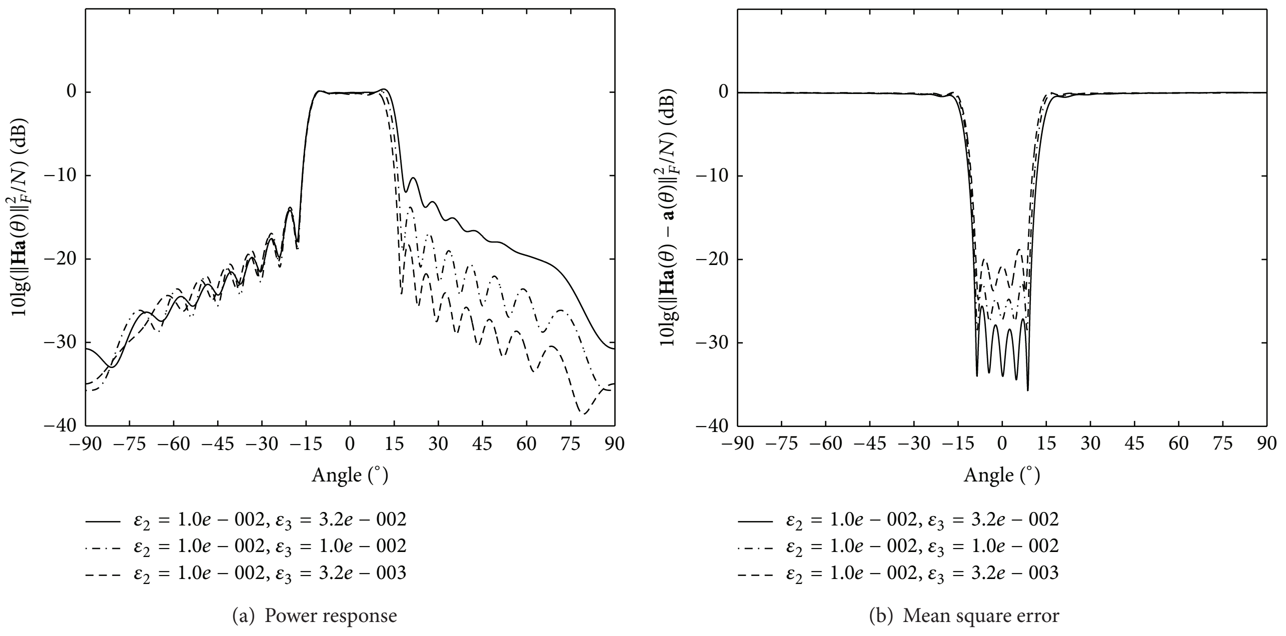


FIGURE 5: Characteristics of the spatial matrix filters designed by DM4.

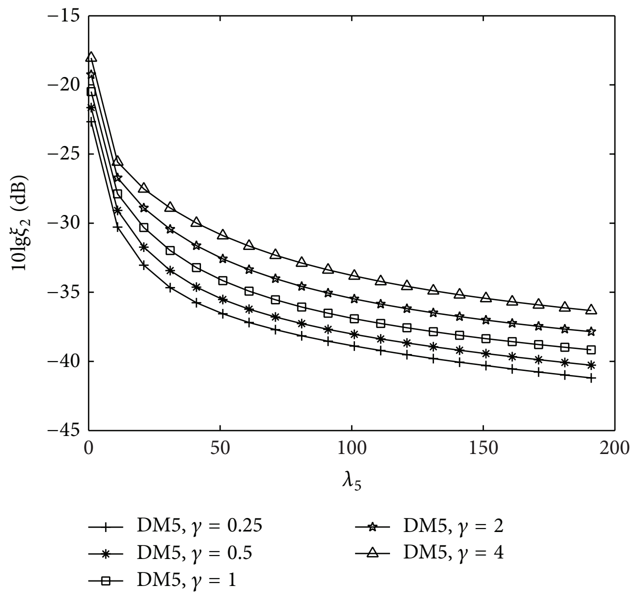


FIGURE 6: The relationship between the optimal Lagrange multiplier $\widehat{\lambda}_5$ and the passband fidelity ξ_2, where the spatial matrix filters were designed by DM5.

simulation, it could be found that the proposed technique was effective for designing spatial matrix filter.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

References

[1] R. J. Vaccaro, A. Chhetri, and B. F. Harrison, "Matrix filter design for passive sonar interference suppression," *The Journal of the Acoustical Society of America*, vol. 115, no. 6, pp. 3010–3020, 2004.

[2] S.-F. Yan and Y.-L. Ma, "Matched field noise suppression: a generalized spatial filtering approach," *Chinese Science Bulletin*, vol. 49, no. 20, pp. 2220–2223, 2004.

[3] S.-F. Yan and Y.-L. Ma, "Optimal design and verification of temporal and spatial filters using second-order cone programming approach," *Science in China F: Information Sciences*, vol. 49, no. 2, pp. 235–253, 2006.

[4] C. S. MacInnes, "Source localization using subspace estimation and spatial filtering," *IEEE Journal of Oceanic Engineering*, vol. 29, no. 2, pp. 488–497, 2004.

[5] D. Han, J. Li, C. Kang, H. Huang, and Q. Li, "Towed line array sonar platform noise suppression based on spatial matrix filtering technology," *Chinese Journal of Acoustics*, vol. 32, no. 4, pp. 379–390, 2013.

[6] D. Han, J. Li, C. Kang, H. Huang, and Q. Li, "Towed line array sonar platform noise suppression based on spatial matrix filtering technique," *Acta Acustica*, vol. 39, no. 1, pp. 27–34, 2014.

[7] R. J. Vaccaro and B. F. Harrison, "Optimal matrix-filter design," *IEEE Transactions on Signal Processing*, vol. 44, no. 3, pp. 705–709, 1996.

[8] Z.-W. Zhu, S. Wang, H. Leung, and Z. Ding, "Matrix filter design using semi-infinite programming with application to DOA estimation," *IEEE Transactions on Signal Processing*, vol. 48, no. 1, pp. 267–271, 2000.

[9] S. Wang, Z.-W. Zhu, and H. Leung, "Semi-infinite optimization technique for the design of matrix filters," in *Proceedings of the 9th IEEE SP Workshop on Statistical Signal and Array Processing*, pp. 204–207, Portland, Ore, USA, September 1998.

[10] D. Han and X. H. Zhang, "Optimal matrix filter design with application to filtering short data records," *IEEE Signal Processing Letters*, vol. 17, no. 5, pp. 521–524, 2010.

[11] D. Han, J. S. Yin, C. Y. Kang, and X. H. Zhang, "Optimal matrix filter design with controlled mean-square sidelobe level," *IET Signal Processing*, vol. 5, no. 3, pp. 306–312, 2011.

[12] S.-F. Yan, C.-H. Hou, and X.-C. Ma, "Matrix spatial prefiltering approach for direction-of-arrival estimation," *Acta Acustica*, vol. 32, no. 2, pp. 151–157, 2007.

[13] X. D. Zhang, *Matrix Analysis and Applications*, Tsinghua University Press, Beijing, China, 2006.

[14] C. F. van Loan, "Generalizing the singular value decomposition," *SIAM Journal on Numerical Analysis*, vol. 13, no. 1, pp. 76–83, 1976.

[15] H. Y. Zha, "The restricted singular value decomposition of matrix triplets," *SIAM Journal on Matrix Analysis and Applications*, vol. 12, no. 1, pp. 172–194, 1991.

Hindawi Publishing Corporation
Journal of Applied Mathematics
Volume 2014, Article ID 181938, 11 pages
http://dx.doi.org/10.1155/2014/181938

Research Article

Semantic Consistency Checking in Building Ontology from Heterogeneous Sources

Shihan Yang,[1,2] Hongyan Tan,[3] and Jinzhao Wu[1,2]

[1] *Guangxi Key Lab of Hybrid Computation and IC Design Analysis, Nanning 530006, China*
[2] *School of Information Sciences and Engineering, Guangxi University for Nationalities (GXUN), Nanning 530006, China*
[3] *Institute of Acoustics Chinese Academy of Sciences, Beijing 100190, China*

Correspondence should be addressed to Shihan Yang; dr.yangsh@gmail.com

Received 16 January 2014; Accepted 8 March 2014; Published 15 April 2014

Academic Editor: X. Song

Semantic collision is inevitable while building a domain ontology from heterogeneous data sources (semi-)automatically. Therefore, the semantic consistency is indispensable precondition for building a correct ontology. In this paper, a model-checking-based method is proposed to handle the semantic consistency problem with a kind of middle-model methodology, which could extract a domain ontology from structured and semistructured data sources semiautomatically. The method translates the middle model into the Kripke structure, and consistency assertions into CTL formulae, so a consistency checking problem is promoted to a global model checking. Moreover, the feasibility and correctness of the transformation is proved, and case studies are provided.

1. Introduction

Semantic web has been a great idea and a promising research area for a dozen years [1]. A main challenge of widening semantic web technologies is lack of semantic data, which is named ontology. So lots of researchers focus on how to transform the legacy mass web data into ontology. The legacy web data in varietal forms can be classified roughly into three types: structured data, such as relational databases; semistructured data, such as XML documents and emails; and nonstructured data, such as general text and video. Recently, technologies of automatically transforming semistructured data or structured data into a domain ontology through mediate modeling are promising [2–6]. In these technologies, some semantic collisions appear inevitably when the same domain ontology has been built from multiple data sources. A domain ontology is a formal expression of a domain knowledge, aiming to unify the general knowledge of a special domain in order to share contents, achieve interoperability, or integrate applications without specific authorization. Unfortunately, the unification is very tough even in the same organization, where the general knowledge exists in very different forms, in very distinct interpretation and in very dissimilar usage for most applications. So automatically creating domain ontology from heterogeneous sources becomes a big challenge. The semantical paradox and ambiguity must be of concern during the process of building the domain ontology, which means the validation of semantic consistency. The semantic consistency guarantees correct and concise specific domain ontology for all kinds of semantic web applications with multiple data sources.

In [2, 7–11], researchers transform structured data (relational database schema) into a middle model and then create a domain ontology from the model. They regard that the relational database is the only data source and that the database is well defined (no ambiguity), which is not always practical. Some semantic preserving properties on transforming are proved, but they do not concern the semantic consistency, which is left for the created domain ontology.

As for semistructured data sources, in [4–6], researchers employ a mediate model language for modeling semistructured data and then transform middle models into the domain ontology. The validity checking has been provided while collisions happen, which is generally a syntax checking. Semantic consistency checking is also missed.

For building domain ontology from heterogeneous data sources, the semantic consistency checking is necessary. In [12, 13], the same middle formal model language has been adopted to model both structured and semistructured data, so the method can be used to build the domain ontology from heterogeneous data sources. In this paper, we focus on the semantic consistency problem based on this method. The literature [14] develops a middle formal language to describe semistructured data for model-checking purpose and [15] employs graph-based formalism to model semistructured data and queries based on the fixed point computation. We are inspired by the model-checking technology [16], translate the mediate model into a Kripke structure, and encode all semantic query problems into CTL formulae and then we transform the semantic consistency checking problem into a global model-checking procedure.

The following Section 2 recalls the mediate-model-based method of building domain ontologies and the model-checking technology. In Section 3, the mediate model language is introduced, and the model equivalence is analyzed. And the model-checking-based consistency checking technology is proposed in Section 4 in detail. Some cases are studied in Section 5. Section 6 gives a conclusion.

2. Mediate-Model-Based Technology and Model Checking

In this section, the method of formally creating domain with the mediate model [12, 13] and a model-checking technology [16, 17] are introduced.

2.1. Formally Creating Domain Ontology. W-graph, a kind of graph-based formal language, is used to model semantic aspect of relationship databases [12]. The main idea is to execute SQL procedures to retrieve the semantic information of the database instances, transform the result sets into a W-graph model, and then transform the model into the ontology autocompletely, which not only maps schemata to the middle model, but also populates the model with data stored in databases. This language is also used to model semantically XML documents in [5]. In [5], they provide XML documents for a semantical interpretation by the W-graph language. The W-graph model created from relationship databases or XML documents can be automatically transformed into a domain ontology (expressed in ontology web language—OWL [18]). And a W-graph is defined as follows.

Definition 1. A W-graph G_w is a directed labeled graph $\langle N, E, \ell\rangle$, where $N = \{N_a, N_c\}$ is a finite set of nodes, N_a is a finite set of *atomic nodes* depicted as ellipses, N_c is a finite set of *composite nodes* depicted as rectangles, $E \subseteq N_c \times (\mathscr{C} \times \mathscr{L}) \times N$ is a set of labeled edges of the form $\langle m, \text{attribute}, n\rangle$, ℓ is defined as $\ell : N \cup E \rightarrow \mathscr{C} \times (\mathscr{L} \cup \{\perp\})$, $\mathscr{C} = \{\text{solid}, \text{dashed}\}$, $\mathscr{L}$ is a set of labels, and $\perp$ is a symbol for nothing (empty label, can be read as bottom).

Nodes in W-graph always represent objects, and edges represent relationships between nodes. There are two types of concrete W-graph: instances and schemata. An instance can be formally defined as follows.

Definition 2. A *W-instance* I is a W-graph such that $\ell_{\mathscr{C}}(e) =$ solid for each edge e of I and $\ell_{\mathscr{C}}(n) =$ solid and $\ell_{\mathscr{L}}(n) \neq \perp$ for each node n of I.

In Figure 1 a W-instance I is depicted.

$$\begin{gathered}
I = \langle N, E, \ell\rangle, \text{ where} \\
N = \langle\{n_1, n_2, n_3, n_4\}, \{n_5, n_6, n_7, n_8\}\rangle, \\
E = \langle(n_1, (\text{solid}, \text{teaches}), n_3), \\
(n_2, (\text{solid}, \text{teaches}), n_3), \\
(n_4, (\text{solid}, \text{attends}), n_3), (n_1, (\text{solid}, \text{age}), n_5), \\
(n_2, (\text{solid}, \text{age}), n_6), (n_3, (\text{solid}, c\text{Name}), n_7), \\
(n_4, (\text{solid}, \text{name}), n_8)\rangle, \\
\ell(n_1) = \ell(n_2) = (\text{solid}, \text{Teacher}), \\
\ell(n_3) = (\text{solid}, \text{Course}), \ell(n_4) = (\text{solid}, \text{Student}), \\
\ell(n_5) = (\text{solid}, 37), \ell(n_6) = (\text{solid}, 40), \\
\ell(n_7) = (\text{solid}, \text{Database}), \ell(n_8) = (\text{solid}, \text{Smith}).
\end{gathered}$$

It describes the information that two teachers, one 37 years old and another 40 years old, teach database course, and the student Smith attends this course. In W-graph, edge attribute consists of two components, the *color* and the *label*, and the function ℓ returns a *color* and a *label* (possibly empty, $\perp$) for each node. Edge labels are stuck close to the corresponding edges, and node labels are written inside the rectangles representing the nodes. The set of colors $\mathscr{C}$ denotes how the lines of nodes and edges are drawn (*solid* or *dashed*), and we also call this information the *color* of a node or edge. On the other hand, the function ℓ can be seen as the composition of the two single valued functions $\ell_{\mathscr{C}}$ and $\ell_{\mathscr{L}}$, so ℓ can also be implicitly defined on edges: if $e = \langle m, \langle c, k\rangle, n\rangle$, then $\ell_{\mathscr{C}}(e) = c$ and $\ell_{\mathscr{L}}(e) = k$. Two nodes may be connected by more than one edge, provided that edge *attributes* are different.

For two subsets of N, S, and T, T is *accessible* from S if for each node n in the set T there is a corresponding node m in the set S such that there exists a path from m to n in W-graph G_w. For example, in the W-instance I of Figure 1, the set $\{n_3, n_5, n_6, n_7, n_8\}$ is accessible from the set $\{n_1, n_2, n_4\}$.

The bisimulation semantics of the language is also given as follows.

Definition 3. Given two W-graphs $G_0 = \langle N_0, E_0, \ell^0\rangle$ and $G_1 = \langle N_1, E_1, \ell^1\rangle$, a relation b is said to be a bisimulation between G_0 and G_1 (write $G_0 \overset{b}{\sim} G_1$) if and only if:

(1) for $i = 0, 1$, $\forall n_i \in N_i$, $\exists n_{1-i} \in N_{1-i}$ such that $n_i b n_{1-i}$,

(2) for $i = 0, 1$, $\forall n_i \in N_i$, $\forall n_{1-i} \in N_{1-i}$, s.t. $n_i b n_{1-i} \rightarrow \ell^i_{\mathscr{L}}(n_i) = \ell^{1-i}_{\mathscr{L}}(n_{1-i}) \vee \ell^i_{\mathscr{L}}(n_i) = \perp \vee \ell^{1-i}_{\mathscr{L}}(n_{1-i}) = \perp$, and

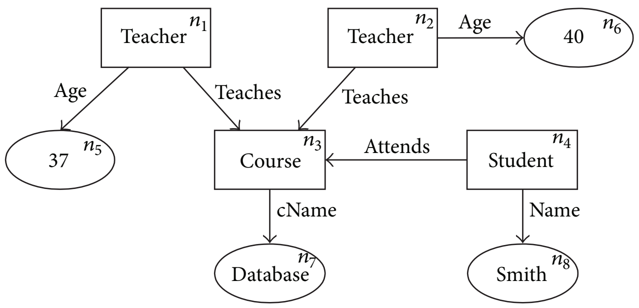


FIGURE 1: A W-instance I.

(3) for $i = 0, 1$, $\forall n \in N_i$, let $M_i(n) \overset{\text{def}}{=} \{\langle m, \text{label}\rangle : \langle n, \langle \text{color}, \text{label}\rangle, m\rangle \in E_i\}$. Then, $\forall n_i \in N_i$, $\forall n_{1-i} \in N_{1-i}$ such that $n_i b n_{1-i}$; for $i = 0, 1$, it holds that $\forall \langle m_i, \ell_i\rangle \in M_i(n_i)$, $\exists \langle m_{1-i}, \ell_{1-i}\rangle \in M_{1-i}(n_{1-i})$, s.t. $m_i b m_{1-i} \wedge \ell_i = \ell_{1-i}$.

2.2. Model Checking. Model checking is an automated technique that, given a finite-state model of a system and a formal property, systematically checks whether this property holds for (a given state in) the model. Here the finite-state model is always called Kripke structure (an automata-like state transition system), and the formal properties are always expressed by computation tree logic (CTL, a logic that is based on a branching-time view) formulae.

Definition 4. A *Kripke structure* $K = \langle \Sigma, \text{Act}, R, I\rangle$ is a transition system over a set Π of atomic propositions, where Σ is a set of states, Act is a set of actions, $R \subseteq \Sigma \times \text{Act} \times \Sigma$ is a transition relations, and $I : \Sigma \rightarrow 2^{\Pi}$ is an interpretation.

A path π in a Kripke structure $K = \langle \Sigma, \text{Act}, R, I\rangle$ is an infinite sequence $\pi = \langle \pi_0, a_0, \pi_1, a_1, \pi_2, a_2, \ldots\rangle$ of states and actions (π_i denotes the ith state in the path π), s.t.. For all $i \in N$ it holds that $\pi_i \in \Sigma$ and either $\langle \pi_i, a_i, \pi_{i+1}\rangle \in R$, with $a_i \in \text{Act}$, or there are no outgoing transitions from π_i and for all $j \geqslant i$ it holds that a_j is the special action τ (which is not in Act) and $\pi_j = \pi_i$.

Definition 5. Given the sets Π and Act of atomic propositions and actions, computation tree logic (CTL) formulae are recursively defined as follows:

(1) each $p \in \Pi$ is a CTL formula;

(2) if ϕ_1 and ϕ_2 are CTL formulae, $a \subseteq \text{Act}$, then $\neg\phi_1$, $\phi_1 \wedge \phi_2$, $\mathsf{AX}_a(\phi_1)$, $\mathsf{EX}_a(\phi_1)$, $\mathsf{AU}_a(\phi_1, \phi_2)$, and $\mathsf{EU}_a(\phi_1, \phi_2)$ are CTL formulae.

A and E are the universal and existential path quantifiers, while neXt (X) and Until (U) are the linear-time modalities. Composition of formulae of the form $\phi_1 \vee \phi_2$, $\phi_1 \rightarrow \phi_2$ can be, respectively, defined by $\neg(\neg\phi_1 \wedge \neg\phi_2)$ and $\neg\phi_1 \vee \phi_2$, and the modalities Finally (F) and Generally (G) can be defined in terms of the CTL formulae: $\mathsf{F}(\phi) = \mathsf{U}(\text{true}, \phi)$ and $\mathsf{G}(\phi) = \neg\mathsf{F}(\neg\phi)$.

Definition 6. Satisfaction of a CTL formula by a state s of the Kripke structure $K = \langle \Sigma, \text{Act}, R, I\rangle$ is defined recursively as follows:

(i) if $p \in \Pi$, then $s \models p$ iff $p \in I(s)$. Moreover, $s \models$ true, and $s \not\models$ false;

(ii) $s \models \neg p$ iif $s \not\models p$;

(iii) $s \models \phi_1 \wedge \phi_2$ iff $s \models \phi_1$ and $s \models \phi_2$;

(iv) $s \models \mathsf{EX}_a(\phi)$ iff there is a path $\pi = \langle s, x, \pi_1, \ldots\rangle$ s.t. $x \in a$ and $\pi_1 \models \phi$;

(v) $s \models \mathsf{AX}_a(\phi)$ iff for all paths $\pi = \langle s, x, \pi_1, \ldots\rangle$, $x \in a$ implies $\pi_1 \models \phi$;

(vi) $s \models \mathsf{EU}_a(\phi_1, \phi_2)$ iff there is a path $\pi = \langle \pi_0, x_0, \pi_1, x_1, \ldots\rangle$, and $\exists j \in \mathbb{N}$ s.t. $\pi_0 = s$, $\pi_j \models \phi_2$, and $\forall i < j$, ($\pi_i \models \phi_1$ and $x_i \in a$);

(vii) $s \models \mathsf{AU}_a(\phi_1, \phi_2)$ iff for all paths $\pi = \langle \pi_0, x_0, \pi_1, x_1, \ldots\rangle$, s.t. $\pi_0 = s$, $\exists j \in \mathbb{N}$, $\pi_j \models \phi_2$, and $\forall i < j$, ($\pi_i \models \phi_1$ and $x_i \in a$).

Definition 7 (the model-checking problem). *Local model-checking*: given a Kripke structure K, a formula ϕ, and a state s of K, the local model-checking is to verify whether $s \models \phi$. *Global model-checking*: given a Kripke structure K, and a formula ϕ, the global model-checking is to find all states s of K such that $s \models \phi$.

If Σ is finite, the global model-checking problem for a CTL formula ϕ can be solved in linear running time on $|\phi| \cdot (|\Sigma| + |R|)$, where $|\phi|$ is the length of formula ϕ and $|\Sigma|$ is the number of elements in the set Σ, $|R|$ is the elements number of the transition relations set R [16].

3. Modeling Semantic Inconsistency

When a language has been used to semantically model different data sources for building the same domain ontology, the semantic collision becomes prominent, so the consistency checking is inevitable. Even from a single data source, the incremental procedure of building the semantical model may fail when a new snippet collides semantically with some model segments that existed. Therefore, ambiguities should be detected in order to get correct semantical model during building a domain ontology. The semantic consistency checking is a mechanism for checking whether the model is semantic unambiguity or paradox.

Two kinds of problems would be of concern: redundancy and paradox. The redundancy-free can reduce the size of the model and accelerate modeling procedure. For the middle-model language W-graph, according to Definition 3, two equivalent models (or model segments) cause a redundancy.

As far as the paradox is concerned, there are four types of inconsistency: concept inconsistency, relationship inconsistency, attribute inconsistency, and fact inconsistencies. Before discussing details of these inconsistencies, another special W-graph, so-called W-schema, will be presented. The schema gives a pattern to organize data for an instance. The schema of W-instance is also a W-graph, and it could be defined formally as follows.

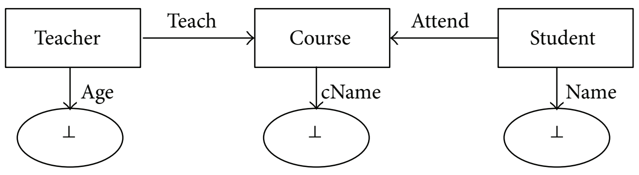


FIGURE 2: A W-schema S of I.

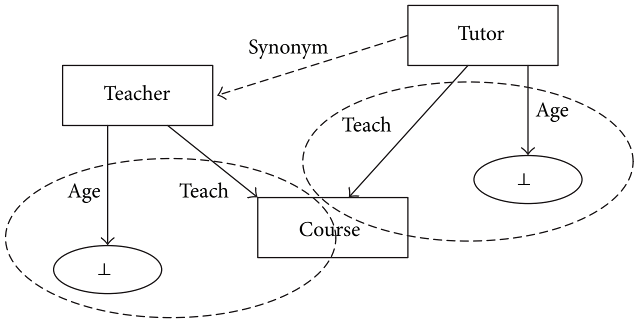


FIGURE 3: A concept redundancy.

Definition 8. A *W-schema S* is a W-graph such that $\ell_{\mathscr{C}}(e)$ = solid for each edge e of S and $\ell_{\mathscr{C}}(n)$ = solid and $\ell_{\mathscr{L}}(n) =\perp$ for each node n of S; that is, a schema has no values.

For example, Figure 2 depicts the W-schema of the W-instance I in Figure 1. So the W-graph language can be used to model semantic aspects of the knowledge, nodes for concepts, edges for relationships between them, W-schemata for the patterns of the knowledge, and W-instances for the concrete contents of that knowledge. Now we will discuss the details of each inconsistency based on the W-graph.

Concept Redundancy. During the procedure of building semantic model, if a new concept occurs, then we add this concept to the model (add a new node to the W-schema). If a concept paradox occurs, then we also add a new concept to the model. But concepts redundancy will occur if we find two concepts (different names, of course) getting the same attribute set and relationship set. Concepts redundancy always occurs in the W-schema. For example, Figure 3 shows that *Teacher* and *Tutor* are the same concepts here. There is not a new concept *Tutor* to be added into the model, but a new synonym of *Teacher* can be annotated by only one edge labeled "synonym."

We define concept redundancy as follows.

Definition 9. In the W-graph $\langle N, E, \ell\rangle$, two concepts C_1 and C_2 (expressed as nodes n_1, n_2) are redundancy when they get the same adjacent nodes set and edges set:

$$C_1 \approx C_2 \text{ iff } |\{n \mid (n, n_1) \in E\}| = |\{m \mid (m, n_2) \in E\}|, \; |\{n \mid (n_1, n) \in E\}| = |\{m \mid (n_2, m) \in E| \text{ and for all } m \text{ and corresponding } n \text{ such that } m \sim n.$$

Here, $m \sim n$ means that after omitting concepts nodes n_1, n_2 and their adjacent edges the subgraph M including node m bisimulates the subgraph N including node n. Computing $m \sim n$ is an iterative process. For deciding whether $m \sim n$, we delete node m and its adjacent edges from M and node n and its adjacent edges from N and then decide whether the two smaller subgraphs are bisimulation. The iterative process will terminate because the nodes and edges are finite.

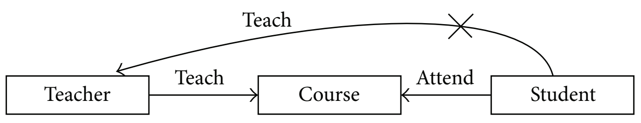


FIGURE 4: A relationship inconsistency.

Relationships Inconsistency. When a new relationship between concepts during modeling process is found, the relationship cannot be added directly into the model, because sometimes the inconsistency will occur. Firstly, if the relationship is redundant according to the bisimulation, it should be ignored. Secondly, if the relationship is added into the model, we must ensure not to introduce some paradox into the model; otherwise, the relationship inconsistency occurs. For example, Figure 4 shows that the relationship "Student teaches Teacher" should cause a paradox. The relationship may be found from XML documents "Teacher teaches students knowledge, and meanwhile teacher also learns something from students." The relationship "teacher learns from students" may be understood as "Student teaches Teacher," but this contradicts "Teacher teaches Student" relationship, which is reasoned from "Teacher teaches Course" and "Student attends the Course."

The paradox relationship can be formally defined as follows.

Definition 10. In the W-graph $\langle N, E, \ell\rangle$, a paradox relationship between node n_1 and node n_2 is that $(n_1, n_2) \in E$ and $(n_2, n_1) \in E$, where $\ell((n_1, n_2)) = \ell((n_2, n_1))$.

Attribute Inconsistency. The attributes of concepts are another kind of relationship, so-called "isa" or "hasa" relationship. So attribute inconsistency is a kind of relationship inconsistency, but simpler.

Fact Inconsistency. This kind of inconsistency occurs in the W-instance. Facts are the individuals of concepts or attributes. When creating W-instance by populating data from the heterogeneous sources, lots of facts inconsistencies may occur. In Figure 5(a), an attribute fact inconsistency will happen if the fact "age is 40" is added into the model, where the teacher named Charley gets two different ages. And in Figure 5(b), the Teacher2 gets all the same value set of attributes, so Teacher1 and Teacher2 are the concept fact inconsistency. Therefore, the two facts "age is 40" and "Teacher2" cannot be added into the model.

The fact inconsistency is formally defined as follows.

Definition 11. In the W-instance $\langle N, E, \ell\rangle$ where $N = \{N_a, N_c\}$, two facts $n_1, n_2 \in N$ are said to be inconsistent if

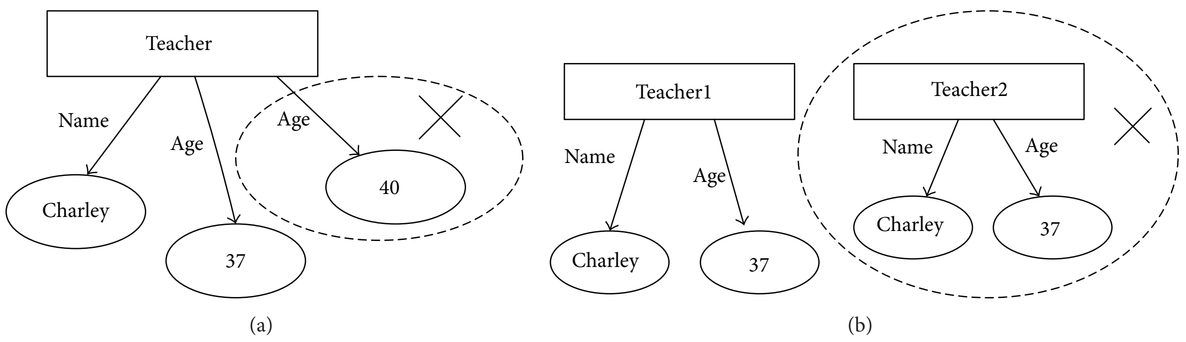


Figure 5: A fact inconsistency.

(1) for $n_1, n_2 \in N_a$, $\exists m \in N_c$ s.t. $\ell((m, n_1)) = \ell((m, n_2))$ and $\ell(n_1) \neq \ell(n_2)$ or

(2) for $n_1, n_2 \in N_c$, $\forall m_1 \in N_a, (n_1, m_1) \in E$ and $m_2 \in N_a, (n_2, m_2) \in E$ s.t. $\ell((n_1, m_1)) = \ell((n_2, m_2))$ and $\ell(m_1) = \ell(m_2)$.

For all these inconsistencies, the core problem is how to discovery the redundancy and paradox in the model. In fact, the procedure of discovery is a subgraph query problem in W-graph.

Definition 12. For a W-graph $G = \langle N, E, \ell \rangle$, a subgraph of G is also a W-graph $G_s = \langle N_s, E_s, \ell_s \rangle$, where $N_s = \{n \in N : \ell_{\mathscr{C}}(n) = \text{solid}\}$ and $E_s = \{(m, (\text{solid}, \ell), n) : m, n \in N_s\}$. Furthermore, given two sets of nodes $S, T \subseteq N$, T is accessible from S if for each $n \in T$ there is a node $m \in S$ such that there is a path in G from m to n.

Definition 13. A W-query is a pointed W-graph, namely, a pair $\langle G, \nu \rangle$ with ν as a node of G (the point). A W-query $\langle G, \nu \rangle$ is accessible if the set N of nodes of G is accessible from $\{\nu\}$.

For example, in Figure 6 three W-queries are depicted. The thick arrows are their *points*, and they are W-queries. Intuitively, the meaning of the first query is to collect all the teachers aged 37. The second query asks for all the teachers that have declared an age (observe the use of an undefined node). The third query, instead, requires collecting all the teachers that teach some courses but not Database, where dashed nodes and lines are introduced to allow a form of negation.

According to Definition 13, an inconsistency checking problem is indeed a query problem in the partial W-graph model. Meanwhile, the semantic equivalence must be of concern during query processing. We call this a *semantical equivalence query problem*. For the incremental procedure of building domain ontology, a semantical equivalence querying should be executed as soon as each new element (concept, relationship, attribute or fact) is added into the model. So, the semantic equivalence querying problem is the basic problem for the consistency checking.

However, the semantical equivalence querying problem for a W-graph model is the subgraph-isomorphism problem, which is NP problem in general. In this paper, we employ model-checking technology to handle semantical equivalence query problem in order to avoid computing subgraph isomorphism.

4. Consistency Checking

In order to employ model-checking technology, we can see a W-graph model as a Kripke structure and a semantical equivalency query as a formula of the temporal logic CTL. In this way, the inconsistency checking problem is reduced to the problem of finding out the states of the model which satisfy the formula (the model-checking problem) that can be done in linear time.

4.1. W-Graph as Kripke Structure.

With the following definition, we can build Kripke structure from W-graph.

Definition 14. Let $G = \langle N, E, \ell \rangle$ be a W-graph, the Kripke structure $K_G = \langle \Sigma_G, \text{Act}_G, R_G, I_G \rangle$ over the set of atomic properties Π_G is defined as follows:

(i) Π_G is the set of all node labels of G, $\Pi_G = \{p \mid \forall n \in N, p = \ell(n)\}$.

(ii) The set of states $\Sigma_G = N$.

(iii) The set of actions $\text{Act}_G = \{p \mid \exists m, n \in N, (m, p, n) \in E\} \cup \{p^{-1} \mid \exists m, n \in N, (m, p, n) \in E\} \cup \{\overline{p} \mid \exists m, n \in N, (m, p, n) \in E\}$; that is, the set of actions includes all the edge labels, negative labels (express as $\overline{p}$), and their inverse labels (express as $p^{-1}, \overline{p}^{-1}$); note that negative labels are very different from inverse labels. A negative label edge expresses that there is no special relationship (i.e., "label" relationship) between two nodes, but an inverse label edge expresses that there is a special relationship (i.e., inverse relationship) between this two nodes.

(iv) The ternary transition relation $R_G = E \cup \{(n, p^{-1}, m) \mid (m, p, n) \in E\}$. Moreover, assume that, for each state s with no outgoing edge in E (a leaf in G), a self-loop edge labeled by the special action τ is added.

(v) The interpretation function $I_G(n) = \{\text{true}, \ell_{\mathscr{L}}(n)\}$, where $n \in N$. That is, in each state n the only formulae that hold are the unary atom $\ell_{\mathscr{L}}(n)$ and true.

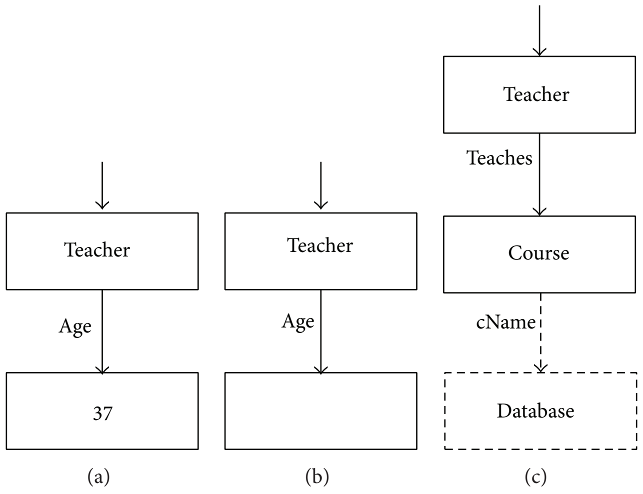


FIGURE 6: Three W-queries.

For instance, consider the W-graph of Figure 1. It holds the following:

(i) Π_G = {Teacher, Course, Student, 37, 40, Databases, Smith},

(ii) $\Sigma_G = \{n_1, n_2, \ldots, n_8\}$,

(iii) $\text{Act}_G = \{\text{age}, \text{age}^{-1}, \overline{\text{age}}, \overline{\text{age}}^{-1}, \text{teaches}, \text{teaches}^{-1}, \overline{\text{teaches}}, \overline{\text{teaches}}^{-1}, \text{attends}, \text{attends}^{-1}, \overline{\text{attends}}, \overline{\text{attends}}^{-1}, \text{name}, \text{name}^{-1}, \overline{\text{name}}, \overline{\text{name}}^{-1}, c\text{Name}, c\text{Name}^{-1}, \overline{c\text{Name}}, \overline{c\text{Name}}^{-1}\}$,

(iv) $R_G = \{(n_1, \text{age}, n_5), (n_5, \text{age}^{-1}, n_1), (n_1, \text{teaches}, n_3), (n_3, \text{teaches}^{-1}, n_1), (n_2, \text{age}, n_6), (n_6, \text{age}^{-1}, n_2), (n_2, \text{teaches}, n_3), (n_3, \text{teaches}^{-1}, n_2), (n_4, \text{attends}, n_3), (n_3, \text{attends}^{-1}, n_4), (n_3, c\text{Name}, n_7), (n_7, c\text{Name}^{-1}, n_3), (n_4, \text{name}, n_8), (n_8, \text{name}^{-1}, n_4), (n_5, \tau, n_5), (n_6, \tau, n_6), (n_7, \tau, n_7), (n_8, \tau, n_8)\}$,

(v) $I_G = \{I_G(n_1) = \{\text{true}, \text{Teacher}\}, I_G(n_2) = \{\text{true}, \text{Teacher}\}, I_G(n_3) = \{\text{true}, \text{Course}\}, I_G(n_4) = \{\text{true}, \text{Student}\}, I_G(n_5) = \{\text{true}, 37\}, I_G(n_6) = \{\text{true}, 40\}, I_G(n_7) = \{\text{true}, \text{Database}\}, I_G(n_8) = \{\text{true}, \text{Smith}\}\}$.

And this Kripke structure is shown in Figure 7.

4.2. Query as CTL Formula. Reviewing the W-query in Figure 6(a), intuitively, the CTL formula must express the statement "the state Teacher formula is true and there is one next state reachable by an edge labeled age, where the 37 formula is true," this is to say

$$\text{Teacher} \wedge \mathsf{EX}_{\text{age}}(37). \tag{1}$$

For the query in Figure 6(b), the CTL formula should be written as

$$\text{Teacher} \wedge \mathsf{EX}_{\text{age}}(\text{true}), \tag{2}$$

For the query of Figure 6(c), we get the following formula:

$$\text{Teacher} \wedge \mathsf{EX}_{\text{teaches}}(\text{Course}) \wedge \mathsf{AX}_{\text{cname}}(\neg\text{Database}). \tag{3}$$

This formula is true if "there is a node labeled Teacher and there exists one next node labeled Course, which is reachable by an edge labeled teaches, such that for all next to Course nodes labeled Database, the relation cName is not fulfilled". Let us then consider how to encode W-queries in CTL formulae. We study the situation that W-graph and W-query are both acyclic. Firstly, we define an auxiliary function to simply handle labels of nodes and edges in W-graph.

Definition 15. Let $G = \langle N, E, \ell \rangle$ be a W-graph, for all nodes $n \in N$ and for all edges $e = \langle n_1, (c, p), n_2 \rangle \in E$; an auxiliary function φ is defined as

$$\varphi(n) = \begin{cases} \ell_{\mathscr{L}}(n), & \text{if } \ell_{\mathscr{L}}(n) \neq \perp, \\ \mathsf{true} & \text{otherwise} \end{cases}$$
$$\varphi(e) = \begin{cases} p, & \text{if } \ell_{\mathscr{C}}(e) = \ell_{\mathscr{C}}(n_2), \\ \overline{p}, & \text{otherwise.} \end{cases} \tag{4}$$

And the query translation can be formally defined as follows.

Definition 16. Let $G = \langle N, E, \ell \rangle$ be a acyclic W-graph, $\nu \in N$, and let $Q = \langle G, \nu \rangle$ be an accessible query. The formula associated with Q is $\Psi_\nu(G)$, where $\Psi_\nu(G)$ is defined recursively as follows:

(i) let $b_1, \ldots, b_h$ $(h \geq 0)$ be the successors of ν, s.t. $\ell_{\mathscr{C}}(b_i) =$ solid,

(ii) let $c_1, \ldots, c_k$ $(k \geq 0)$ be the successors of ν, s.t. $\ell_{\mathscr{C}}(c_i) =$ dashed,

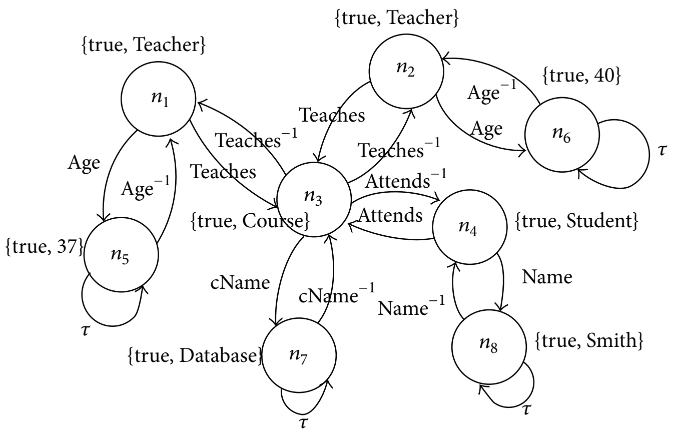


FIGURE 7: The Kripke structure of Figure 1.

(iii) for $i = 1, \ldots, h$ and $j = 1, \ldots, k$, let e_i be the edge which links v to b_i and e_i' the one which links v to c_j. If $\ell_{\mathscr{C}}(v) = \text{solid}$, then

$$\begin{aligned}\Psi_v(G) &= \varphi(v) \wedge \bigwedge_{i=1,\ldots,h} \mathsf{EX}_{\varphi(e_i)}\left(\Psi_{b_i}(G)\right) \\ &\wedge \bigwedge_{j=1,\ldots,k} \mathsf{AX}_{\varphi(e_j')}\left(\Psi_{c_j}(G)\right) \\ \text{else } &\left(\ell_{\mathscr{C}}(v) = \text{dashed}\right) \Psi_v(G) \\ &= \neg\varphi(v) \vee \bigvee_{i=1,\ldots,h} \mathsf{AX}_{\varphi(e_i)}\left(\Psi_{b_i}(G)\right) \\ &\vee \bigvee_{j=1,\ldots,k} \mathsf{EX}_{\varphi(e_j')}\left(\Psi_{c_j}(G)\right).\end{aligned} \tag{5}$$

The construction of the formula involves the unfolding of a directed acyclic graph; the size of the formula $\Psi_v(G)$ (written as $|\Psi_v(G)|$) can grow exponentially with respect to$|G|$. Although that, it is easy to compute the formula without repetitions of subformulae and keep the memory allocation linear w.r.t. $|G|$, so it is natural to compactly represent the formula using a linear amount of memory. This compact representation is allowed by the model-checker NuSMV [19].

Theorem 17. *Given a W-instance I and a W-query $\langle G, v\rangle$, let K_I be the Kripke structure associated with I and $\Psi_v(G)$ the CTL formula associated with $\langle G, v\rangle$. Consistency checking I with $\langle G, v\rangle$ can be done in linear time on $|I| \cdot |\Psi_v(G)|$.*

Proof. For achieving the consistency checking procedure, there are three steps to follow, translating the W-graph into a Kripke structure, encoding the W-query into a CTL formula, and executing a model-checking process.

Assuming that $|I| = |N| + |E|$, where $|N|$ is the number of nodes in W-instance I and $|E|$ is the number of edges in this graph, notating $K_I = \langle \Sigma_I, \text{Act}_I, R_I, I_I\rangle$, expressing $|\Sigma_I|, |\text{Act}_I|, |R_I|$ as the number of states, actions, and transition relations in $|K_I|$, and writing $|\Psi_v(G)|$ as the length of the CTL formula $\Psi_v(G)$, then the complexity issues can be analyzed.

Firstly, when creating the Kripke structure, for each node, action, and transitional relation that must be created one by one, the time complexity should be $|\Sigma_I| + |\text{Act}_I| + |R_I| \leq |N| + 4|E| + 3|E| = O(|I|)$, according to Definition 14. Secondly, we consider the length of the encoded CTL formula $|\Psi_v(G)| \leq |G| = O(|G|)$, because the worst situation is to encode the whole W-graph into a formula, where $|G|$ is the total number of edges and nodes in this query. The last step is to execute model-checking procedure; the time complexity is $|\Psi_v(G)| \cdot (|\Sigma_I| + |R_I|) = O(|G| \cdot |I|)$ according to [16]. So the total time complexity is $O(|I|) + O(|G|) + O(|G| \cdot |I|)$, which is a linear time on $|I| \cdot |\Psi_v(G)|$.

If we think that the $|G|$ is always less than or equal to $|I|$ (intuitively, it is always so), then $O(|I|) + O(|G|) + O(|G| \cdot |I|) \leq O(|I|^2) + O(|I|)$, which is a polynomial time over $|I|$. □

The equivalence between two segments of W-graph is the equivalence between two CTL formulae, which can be formally proved in linear time. The consistency checking problem for the W-graph model is promoted to the equivalence proof problem for the CTL formulae with a Kripke structure.

4.3. Checking Consistency. During the procedure of extracting a domain ontology gradually from heterogeneous data sources, semantic consistency has to be checked. According to Definitions 14 and 16, after encoding a W-graph to a Kripke structure and a W-query to a CTL formula, the semantic equivalent query problem on the W-graph has been promoted to a global model-checking problem.

Definition 18. Given a W-instance I and a W-query $\langle G, v\rangle$, let K_I be the Kripke structure associated with I and $\Psi_v(G)$ the CTL formula associated with $\langle G, v\rangle$. The Consistency checking I with $\langle G, v\rangle$ amounts to solve the global model-checking problem with the Kripke structure K_I and the CTL formula $\Psi_v(G)$, namely, to find all the states s of K_I such that $s \vDash \Psi_v(G)$. Algorithm 1 describes this procedure.

```
Require: $I$, // W-graph semantic model
    $\langle G, v\rangle$ // semantic segment expressed by W-query
Ensure: consistent or inconsistent
    encoding $I$ to $K_I$; // according to Definition 14
    encoding $\langle G, v\rangle$ to $\Psi_v(G)$; // according to Definition 16
    $S = \{s \mid s \vDash \Psi_v(G)\}$; // model checking
    if $S$ is emptyset then
        return consistent;
    else
        return inconsistent;
    end if
```

Algorithm 1: Semantic consistency checking.

So semantic consistency checking can be done with model-checking technology. Let us consider each type of inconsistency defined above. As for concepts inconsistency, it is to find out whether there is another concept (equivalent to C) in W-schema S, which has been built to be extended, when a new concept C has been added into the W-schema S. This is to say, a W-query $Q = \langle S, C\rangle$ should be imposed on the S for consistency checking. Let K_S be the Kripke structure of S and let $\Psi_C(S) = \varphi(C)$ be the CTL formula of Q and we should find all states s of K_S such that $s \vDash \varphi(C)$.

As far as relationship inconsistency is concerned, it is to query a W-graph G with a W-query $Q = \langle G, n_1\rangle$ before a new relationship (n_1, n_2) is added into the W-graph model. Let K_G be the Kripke structure of G, and the CTL formula is

$$\Psi_{n_1}(G) = \varphi(n_1) \wedge \mathsf{EX}_{\varphi((n_1,n_2))}(n_2). \tag{6}$$

The consistency checking is to find all states s of K_G such that $s \vDash \Psi_{n_1}(G)$. If the states like this cannot be found, then the model is consistent after adding the new relationship (n_1, n_2); if any state has been found, then the model will be inconsistent and the new relationship cannot be added into the model. This is the relationship redundant inconsistency. As for paradox relationship paradox inconsistency, the CTL formula is

$$\Psi_{n_2}(G) = \varphi(n_2) \wedge \mathsf{EX}_{\varphi((n_1,n_2))}(n_1). \tag{7}$$

As for fact consistency checking, the Kripke structure of the W-instance I is K_I, and the CTL formula for the attribute fact (the instance m of a concept has the concrete attribute value n, i.e., (m, n)) inconsistency is

$$\Psi_m(I) = \varphi(m) \wedge \mathsf{EX}_{\varphi((m,n))}(n). \tag{8}$$

The CTL formula for the concept fact (the instance m of one concept to be added into the model) inconsistency is

$$\Psi_m(I) = \varphi(m) \wedge \bigwedge_{j=1,\ldots,k} \mathsf{AX}_{\varphi((m,n_j))}\left(\Psi_{n_j}(I)\right), \tag{9}$$

where n_j, $j = 1, \ldots, k$ are all successors of m. If some states of K_I satisfy the above formula, then the inconsistency occurs. This is to say, a fact m cannot be added into the model if an equivalent fact has already existed in the model.

5. Cases Study

In this section, we will test the semantic consistency checking technique presented above by using the model checker NuSMV 2.5.4 [20]. NuSMV allows for the representation of finite-state machines (FSMs) and for the analysis of specifications expressed in computation tree logic (CTL) using symbol model-checking techniques. For using the tool, we first rewrite the Kripke structure into a finite-state machine, where edges are not labeled, as follows:

given a Kripke structure related to a W-instance, replace every labeled edge $m \xrightarrow{\text{action}} n$ by the two edges $m \rightarrow \mu$, $\mu \rightarrow n$, where μ is a new node labeled action.

The input of the NuSMV tool is represented by a SMV program, which can express both the FSMs and CTL formulae. The Algorithm 2 is the SMV program to describe the Kripke structure of the W-instance in Figure 7 and some CTL specification of consistency checking mentioned above, where the line started with symbol "- -" is comment.

In this SMV program, the set of states of the Kripke structure is chosen by declaring the state variable `state` to assume values {`n0`, ..., `n8`, `teaches`, ..., `tao`}, where actions have also been seen as states. The transition relation of the Kripke structure is expressed by assigning (`ASSIGN`), for each value of the variable `state`, the list of nodes that can be reached from it through one edge. The variables `label` and `nl` are introduced to define the node label of each state identified by the value of the variable `state`. And CTL formulae have been defined by `CTLSPEC`. The formula

`(nl = Teacher) & EX(state = age) & EX(state = teaches & EX(nl = Course))`

says when a new concept *Teacher*, which has an *age* attribute, has a teaches action, and can teach some Course, is to be added into the model whether there exists an inconsistency. And

`(nl = Student) & EX(state = attends & EX(nl = Course))`

expresses whether a new relationship Student $\xrightarrow{\text{attends}}$ Course can be added into the model without semantic inconsistency. And

`(nl = Teacher) & EX(state = age & EX(nl = 40))`

says whether the attribute relationship "the Teacher is 40 years old" can be added into the model without any redundancy and paradox. And

`(nl = Teacher) & EX(state = age & EX(nl = 37)) & EX(state = teaches & EX(nl = Course & EX(state = cName & EX(nl = Database))))`

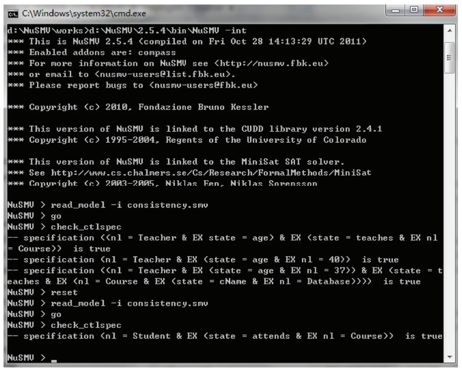
expresses whether the fact "the 37-year Teacher teaches Database Course" can be added into the model.

The results we have obtained on the 64-bit windows 7 operation system are shown in Figure 8. The output *true* for a

```
-- SMV program for consistency checking
MODULE main
  VAR
    state:{n1,n2,n3,n4,n5,n6,n7,n8,teaches,inv_teaches,attends,
      inv_attends,age,inv_age,name,inv_name,cName,inv_cName,tao};
    label:{Teacher,Course,Student,Database,Smith,37,40};
  ASSIGN
    init(state) := n1;
    next(state) := case
      state = n1 | state = n2 : {teaches,age};
      state = teaches : n3;
      state = age : {n5,n6};
      state = n3:{inv_attends,inv_teaches,cName};
      state = inv_attends : n4;
      state = inv_teaches : {n1,n2};
      state = cName : n7;
      state = n4 : {attends,name};
      state = attends : n3;
      state = name : n8;
      state = n5 : {inv_age,tao};
      state = inv_age : {n1,n2};
      state = tao : {n5,n6,n7,n8};
      state = n6 : {inv_age,tao};
      state = n7 : {inv_cName,tao};
      state = inv_cName : n3;
      state = n8 : {inv_name,tao};
      state = inv_name : n4;
      TRUE : state;
    esac;
  DEFINE
    nl := case
      state = n1 | state = n2 : Teacher;
      state = n3 : Course;
      state = n4 : Student;
      state = n5 : 37;
      state = n6 : 40;
      state = n7 : Database;
      state = n8 : Smith;
      TRUE : state;
    esac;
  -- concept redundancy checking
  CTLSPEC (nl=Teacher) & EX(state=age) &
          EX(state=teaches & EX(nl=Course))
  -- relationship inconsistency checking
  -- need to be changed to 'init(state) := n4;'
  CTLSPEC (nl=Student) & EX(state=attends & EX(nl=Course))
  -- attribute inconsistency checking
  CTLSPEC (nl=Teacher) & EX(state=age & EX(nl=40))
  -- fact inconsistency checking
  CTLSPEC (nl=Teacher) & EX(state=age&EX(nl=37)) &
          EX(state=teaches & EX(nl=Course &
          EX(state=cName & EX(nl=Database))))
```

Algorithm 2

CTL formula says that at least one inconsistency exists when checking the W-graph model with the new semantic segment (expressed by this formula), so the new segment cannot be added into the model. Otherwise, we can choose another initiate state to check again, until we finish all elements of initiated state set. If we always get a final *false* output, then the inconsistency has not occurred, and the new semantic segment can be added into the model.


C:\Windows\system32\cmd.exe

```
d:\NuSMV\works>d:\NuSMV\2.5.4\bin\NuSMV -int
*** This is NuSMV 2.5.4 (compiled on Fri Oct 28 14:13:29 UTC 2011)
*** Enabled addons are: compass
*** For more information on NuSMV see <http://nusmv.fbk.eu>
*** or email to <nusmv-users@list.fbk.eu>.
*** Please report bugs to <nusmv-users@fbk.eu>

*** Copyright (c) 2010, Fondazione Bruno Kessler

*** This version of NuSMV is linked to the CUDD library version 2.4.1
*** Copyright (c) 1995-2004, Regents of the University of Colorado

*** This version of NuSMV is linked to the MiniSat SAT solver.
*** See http://www.cs.chalmers.se/Cs/Research/FormalMethods/MiniSat
*** Copyright (c) 2003-2005, Niklas Een, Niklas Sorensson

NuSMV > read_model -i consistency.smv
NuSMV > go
NuSMV > check_ctlspec
-- specification ((nl = Teacher & EX state = age) & EX (state = teaches & EX nl
= Course))  is true
-- specification (nl = Teacher & EX (state = age & EX nl = 40))  is true
-- specification ((nl = Teacher & EX (state = age & EX nl = 37)) & EX (state = t
eaches & EX (nl = Course & EX (state = cName & EX nl = Database))))  is true
NuSMV > reset
NuSMV > read_model -i consistency.smv
NuSMV > go
NuSMV > check_ctlspec
-- specification (nl = Student & EX (state = attends & EX nl = Course))  is true

NuSMV > _
```

Figure 8: Experiment results.

This test confirms the possibility of solving semantic consistency checking problem by using model-checking on polynomial time methods.

6. Conclusion

For validating semantic consistency during the increasing procedure of building a domain ontology from heterogeneous sources, we employ the model-checking technology to avoid subgraph isomorphism problem, which is NP hard. In order to adopt model-checking method, we formally transform the semantic model into a Kripke structure and the semantic equivalent querying problem into CTL formulae and then the semantic consistency is promoted to the global model-checking problem. The effective experiment with the model-checking tool NuSMV has also been introduced. In the future, the reasoning problem should be considered clearly; for example, some implicative semantic elements would be reasoned from the existing model. If a new semantic segment is equivalent to some implicative semantic elements, the inconsistency also occurs. In the near future, this type of consistency checking should also be regarded.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

This paper is supported by the Natural Science Foundation of Guangxi under Grants nos. 2011GXNS-FA018154 and 2012GXNS-FGA060003, the Science and Technology Foundation of Guangxi under Grant no. 10169-1, Guangxi Scientific Research Project no. 201012MS274, and the starting fund of GXUN under Grant no. 2011QD017. This paper is supported also by Grant (2012HCIC04) of Guangxi Key Laboratory of Hybrid Computation and IC Design Analysis Open Fund.

References

[1] T. Berners-Lee, J. Hendler, and O. Lassila, "The semantic web," *Scientific American*, vol. 284, no. 5, pp. 34–43, 2001.

[2] J. Barrasa, Ó. Corcho, and A. Gómez-pérez, "R2o, an extensible and semantically based database-to-ontology mapping language," in *Proceedings of the 2nd Workshop on Semantic Web and Databases (SWDB '04)*, pp. 1069–1070, Springer, 2004.

[3] F. Zhang, L. Yan, Z. M. Ma, and J. Cheng, "Knowledge representation and reasoning of XML with ontology," in *Proceedings of the 26th Annual ACM Symposium on Applied Computing (SAC '11)*, pp. 1705–1710, New York, NY, USA, March 2011.

[4] T. Pankowski, "Detecting semantics-preserving xml schema mappings based on annotations to owl ontology," in *Proceedings of the 4th International Workshop on Logic in Databases (LID '11)*, pp. 57–57, New York, NY, USA, 2011.

[5] S. Yang, J. Wu, A. He, and Y. Rao, "Derivation of owl ontology from xml documents by formal semantic modeling," *Journal of Computers*, vol. 8, no. 2, pp. 372–379, 2013.

[6] S. Yang and J. Wu, "Mapping relational databases into ontologies through a graph-based formal model," in *Proceedings of the 6th IEEE International Conference on Semantics Knowledge and Grid (SKG '10)*, pp. 219–226, 2010.

[7] C. Bizer, "D2r map—a database to rdf mapping language," in *Proceedings of the 12th ACM International World Wide Web Conference (WWW '03)*, Budapest, Hungary, 2003.

[8] E. Dragut and R. Lawrence, "Composing mappings between schemas using a reference ontology," in *On the Move to Meaningful Internet Systems 2004: CoopIS, DOA, and ODBASE*, pp. 783–800, Springer, Berlin, Germany, 2004.

[9] D. Dejing, L. Paea, K. Shiwoong, and Q. Peishen, "Integrating databases into the semantic web through an ontology-based framework," in *Proceedings of the 22nd International Conference on Data Engineering Workshops (ICDEW '06)*, IEEE Computer Society, p. 54, Washington, DC, USA, 2006.

[10] A. Yuan, B. Alex, and M. John, "Inferring complex semantic mappings between relational tables and ontologies from simple correspondences," in *On the Move to Meaningful Internet Systems 2005: CoopIS, DOA, and ODBASE*, pp. 1152–1169, Springer, Berlin, Germany, 2005.

[11] H.-H. Do and E. Rahm, "Coma: a system for flexible combination of schema matching approaches," in *Proceedings of the 28th International Conference on Very Large Data Bases (VLDB '02)*, pp. 610–621, VLDB Endowment, 2002.

[12] S. Yang, Y. Zheng, and X. Yang, "Semi-automatically building ontologies from relational databases," in *Proceedings of the 3rd IEEE International Conference on Computer Science and Information Technology (ICCSIT '10)*, vol. 1, pp. 150–154, July 2010.

[13] S. H. Yang, J. Z. Wu, and A. P. He, "Automatically transforming legacy xml documents into owl ontology," *Applied Mechanics and Materials*, vol. 241, pp. 2638–2644, 2013.

[14] A. Dovier and E. Quintarelli, "Applying model-checking to solve queries on semistructured data," *Computer Languages, Systems and Structures*, vol. 35, no. 2, pp. 143–172, 2009.

[15] S. Cluet, "Modeling and querying semi-structured data," in *Information Extraction a Multidisciplinary Approach to an Emerging Information Technology*, M. Pazienza, Ed., vol. 1299 of *Lecture Notes in Computer Science*, pp. 192–213, Springer, 1997.

[16] E. M. Clarke, O. Grumberg, and D. A. Peled, *Model Checking*, MIT Press, 1999.

[17] C. Baier and J.-P. Katoen, *Principles of Model Checking*, MIT Press, Cambridge, Mass, USA, 2008.

[18] B. Motik, P. F. Patel-Schneider, and B. Parsia, "Owl 2 web ontology language structural specification and functional-style syntax, W3Cr," 2009, http://www.w3.org/TR/2009/REC-owl2-syntax-20091027/.

[19] A. Cimatti, E. Clarke, F. Giunchiglia, and M. Roveri, "Nusmv: a new symbolic model checker," *International Journal on Software Tools for Technology Transfer*, vol. 2, no. 4, pp. 410–425, 2000.

[20] C. Alessandro, R. Marco, C. Roberto et al., "Nusmv2 (version 2.5.4)," 2011, http://nusmv.fbk.eu/.

Hindawi Publishing Corporation
Journal of Applied Mathematics
Volume 2014, Article ID 645235, 10 pages
http://dx.doi.org/10.1155/2014/645235

Research Article

Test Purpose Oriented I/O Conformance Test Selection with Colored Petri Nets

Jing Liu, Xinming Ye, and Jiantao Zhou

College of Computer Science, Inner Mongolia University, Hohhot 010021, China

Correspondence should be addressed to Jing Liu; liujing@imu.edu.cn

Received 28 January 2014; Accepted 14 March 2014; Published 14 April 2014

Academic Editor: Guiming Luo

This paper proposes an input-output conformance (IOCO) test selection method directed by test purpose model specified with colored Petri nets (CPN). Based on conformance testing oriented CPN models for specifying software functional behaviors and specific test purposes, respectively, feasible test cases are generated, guided by the CPN based IOCO relation, using synchronized model simulation with the proof of the soundness of test generation and the coverage towards test purposes. This test selection method integrates the merits the IOCO testing theory and the CPN modeling synergistically and is applied as a novel and applicable test selection method for actual testing practice of large-scale software systems. As the synchronized model simulation with two CPN models is irrespective of their model scale, the effectiveness and practicability of our test selection method are enhanced with scalability.

1. Introduction

Software systems running based on network environment are ubiquitous. It is quite significant to validate their functional correctness. Conformance testing [1] just aims at checking whether the software implementation conforms to its functional specification. Therefore, conformance testing is indispensable in such software system validation process. Nowadays, model based testing (MBT) technology is introduced and well developed to promote the efficiency and effectiveness of conformance test generation [2–6]. It allows for generating test cases with test oracles from a formal model that specifies the software behaviors explicitly, which improves the low-level efficiency and inaccuracy of the manual test case generation process. In particular, concerning the conformance testing towards network based software systems, the well-established input-output conformance (IOCO) testing theory and technologies [6–9] are more feasible, because the IOCO relation formally defines what external output should be observed through the practical test execution and how to determine the conformance based on these observations. In our studies, network based software systems are adopted as our system under testing (SUT) and the IOCO testing theory is the most significant theoretical foundation in our testing research.

In original IOCO testing theory, labeled transition system (LTS) is utilized as its basic formal models. However, compared with LTS or other formal modeling methods such as automata or process algebra, colored Petri nets (CPN) [10] have more advantages for specifying and validating complicated functional behaviors of network software systems. First, CPN could not only specify the detailed and complicated software functionalities intuitively and hierarchically but also support visible simulation and efficient analysis to validate the correctness of software behaviors. Validated models are indispensable for successful application of MBT approaches. Second, CPN models can execute dynamically, which is directed by the data-dependent control flow of system behaviors. Generating test cases by such model simulation process, they certainly contain actual test data and test oracles, so they are quite feasible for guiding practical test execution.

We have proposed the introductory idea of CPN model based IOCO test generation approach in our conference paper [11]. However, that paper just focuses on elementary test case generation approach, and, in this paper, we focus on test case selection; that is, test purpose is considered in test case generation to improve its pertinence, and a novel test purpose model oriented IOCO test selection method is proposed. Specifically, conformance testing oriented colored

Petri nets (CT-CPN) models are used as formal models for modeling software specification, and PN-ioco relation is defined [11]. Then, we model test purposes as CT-CPN models, and test cases are generated using synchronized model simulation between a specification CT-CPN model and a test purpose CT-CPN model. Besides, we prove the soundness of test generation; that is, as long as the implementation fails one test case, it will definitely not conform to its specification. We also prove the coverage towards test purpose; that is, generated test cases should cover and only cover functional behaviors which are specified in test purpose models. We finally apply the test selection method into a file sharing software system to illustrate its feasibility and effectiveness.

CPN model based IOCO test selection has several advantages, compared with current IOCO test selection method in literatures [12–14]. First, CPN models can execute dynamically, which is directed by the data-dependent control flow of system behaviors. Generating by such model simulation process, test cases certainly contain actual test data and test oracles, so they are quite feasible for guiding practical test execution. Second, as the synchronized model simulation with two CPN models is irrespective of their model scale, the effectiveness and practicability of our test selection method are enhanced with scalability. In a word, a CPN model based IOCO test selection method tends to be a promising testing technology to validate the correctness of reactive network software systems more efficiently and more effectively.

The paper is organized as follows. The related work and preliminaries are discussed in Section 2. The framework of our CPN model based IOCO test selection method is introduced in Section 3. The formal definition of CT-CPN models and PN-ioco relation is recalled in Section 4 as basic knowledge. Then, in Section 5, a novel test case selection algorithm is proposed using synchronized simulation technology in CPN modeling context to guarantee that all test cases are feasible for practical test execution and totally cover test purposes. In Section 6, we prove the soundness of test generation and the coverage degree towards test purposes. As a representative, we apply the test selection method into a file sharing software system and perform its actual test selection and execution procedure to illustrate the feasibility and effectiveness of our test selection method.

2. Related Work and Preliminaries

As for test case generation approaches based on CPN models, Watanabe and Kudoh [15] propose a basic test generation algorithm, which could be considered as the first step in this field. First, the reachability tree of a CPN model is constructed, and all input-output sequences from the root node to leaf nodes in this tree are traversed to form the test cases, and, then, equivalent markings in that tree are combined to construct the corresponding reachability graph, and FSM model based test case generation approaches are applied directly based on this graph. Recently, Farooq et al. [16] use random walking technology to randomly traverse the model state space to generate the test cases, where several sequential coverage criteria and concurrent coverage criteria are proposed to guide the test selection. Zhu and He [17] have proposed four specific testing strategies towards the high-level Petri nets. For each strategy, they first define a set of schemes to observe and record the testing results, and they also define a set of coverage criteria to measure the test adequacy. But, no detailed test generation algorithms are explicitly presented. We have proposed the introductory idea of CPN model based IOCO testing approach [11]. It focuses on elementary test case generation approach, and, in this paper, we focus on test selection method.

In order to promote the pertinence of testing projects, certain test selection criterion is always adopted in test generation process to produce finite and indispensable actual test cases. It is quite propitious for performing feasible testing projects under constraint of testing execution time and cost. Generally, test selection for function testing is classified as test purposed oriented methods [9, 12, 13, 18], random testing methods [8, 19, 20], property coverage based methods [21, 22], and symbolic test data selection methods [14]. However, test purposed oriented methods usually specify part of functional behaviors of software as test purpose model and make generated test cases focus on testing such specific behaviors. It is quite propitious for performing feasible conformance testing towards network based software systems with black-box and reactive behavior characteristics. In this paper, a novel IOCO test selection method is proposed using synchronized model simulation technology between two CPN models. It has high scalability for dealing with the larger software models.

CPN is advantaged for modeling and validation of systems where concurrency and communication are key characteristics. Its formal definitions are referred to in [10]. Besides, other key definitions concerning the behavior simulation of CPN models which are used in following sections are listed as follows.

Definition 1. For a CPN = $(P, T, A, \Sigma, V, C, G, E, I)$, consider

(1) *preset* and *postset* of the place or the transition:

$$\begin{aligned} &\forall p \in P : \operatorname{pre}(p) = \{t \in T \mid (t, p) \in A\}; \\ &\operatorname{post}(p) = \{t \in T \mid (p, t) \in A\}, \\ &\forall t \in T : \operatorname{pre}(t) = \{p \in P \mid (p, t) \in A\}; \\ &\operatorname{post}(t) = \{p \in P \mid (t, p) \in A\}; \end{aligned} \tag{1}$$

(2) $M \overset{\sigma}{\Rightarrow} =_{\text{def}} \exists M_i : M \overset{\sigma}{\Rightarrow} M_i$, where $\sigma = (t_0, b_0), (t_1, b_1) \cdots (t_{i-1}, b_{i-1})$, $M \xrightarrow{(t_0,b_0)} M_1 \xrightarrow{(t_1,b_1)} \cdots \xrightarrow{(t_{i-1},b_{i-1})} M_i$, if $|\sigma| = 1$, $M \overset{\sigma}{\rightarrow}$ is used instead;

(3) $\operatorname{trace}(M) =_{\text{def}} \{\sigma \in \operatorname{BE}(T)^* \mid M \overset{\sigma}{\Rightarrow}\}$;

(4) M fires $\sigma =_{\text{def}} \{M_n \mid M \overset{\sigma}{\Rightarrow} M_n, \sigma \in \operatorname{BE}(T)^*\}$;

(5) CPN is *deterministic*, if $|M \text{ fires } \sigma| \leq 1$;

(6) CPN has *finite output*, if $|M \text{ fires } \sigma| \leq n$ $(n \in N)$;

(7) CPN has *finite behavior*, if $\exists n \in N, \forall \sigma \in \operatorname{trace}(M_0) : |\sigma| < n$.

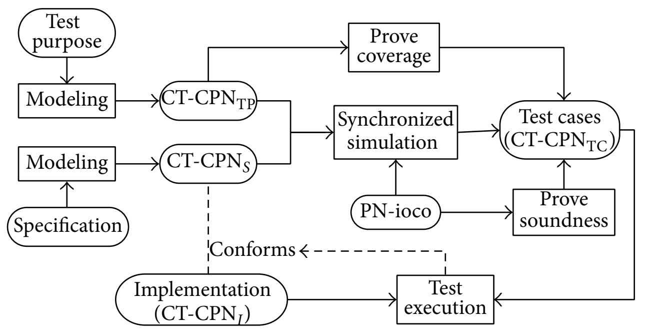


Figure 1: The framework of IOCO test selection method with CPN models.

In this definition, M stands for markings, σ stands for occurrence trace of system behavioral execution, and BE stands for binding elements with a transition and its value bindings into variables.

3. Methodology Overview

Integrating the merits of the IOCO testing theory and the CPN modeling synergistically and then constructing a novel IOCO test selection method based on test purpose CPN models are the major research goals of this paper. However, simply replacing LTS modeling with CPN modeling does not make sense. In Figure 1, we propose a framework of IOCO test selection methodology based on CPN models, which is composed of three related parts in the whole test selection process.

First, modified CPN modeling is proposed as CT-CPN models to specify key characteristics and requirements for conformance testing scenario accurately. For example, CT-CPN$_S$ models software functional behaviors according to software requirement specification, CT-CPN$_I$ models actual software implementation behaviors, CT-CPN$_{TP}$ models software functional behaviors of specific test purposes, and CT-CPN$_{TC}$ models finally generated test cases. Such CT-CPN series models explicitly specify external visible actions which are significant in practical test execution, that is, to make the most of both the place and the transition elements in CT-CPN models to distinguish visible actions from internal actions. In particular, to deal with the special output actions, such as the quiescence or deadlock [7], it introduces new kind of transitions to model them accurately. Besides, a corresponding implementation relation in the context of CT-CPN modeling is proposed as PN-ioco relation to precisely specify what it means for an implementation to conform to its functional specification. In CPN model context, software behaviors are simulated with specific system data, so we have to determine the IOCO conformance via comparing the output actions with specific data. This part of contents had been proposed in our conference paper [11], and, in order to make this paper self-contained, we will show the basic formal definition of CT-CPN model and PN-ioco relation in Section 4.

Second, based on the above CT-CPN modeling and PN-ioco relation, we need to develop a novel test selection approach with two desired goals. One is to make the test selection with high scalability for dealing with more complicated system models, and the other one is to make all test cases feasible for the practical test executions. Therefore, in Section 5, we model test purposes as CT-CPN$_{TP}$ models and then propose a novel IOCO test case selection method, where test cases are generated using synchronized model simulation between a CT-CPN$_S$ model and a CT-CPN$_{TP}$ model. When the synchronized simulation procedure terminated, a final test case is produced and specified as a CT-CPN$_{TC}$ model. As the synchronized model simulation is irrespective of model scale, its effectiveness and practicability tend to be enhanced with high scalability.

Third, in Section 6, we prove the soundness of test generation; that is, as long as the software implementation fails one test case, it will definitely not conform to its functional specifications. We also prove the coverage towards test purposes; that is, generated test cases should cover and only cover functional behaviors which are specified in test purpose models. In this way, under constraint of testing execution time and cost, we could generate finite and indispensable actual test cases to promote the pertinence of testing projects.

Developing a CPN model based IOCO test selection method is challenging but quite promising. It has solid theoretical foundation and bright application prospect, so it tends to be used as a competent and effective conformance testing technology to validate the correctness of network based software systems.

4. CT-CPN Modeling and PN-ioco Relation

4.1. Specification Modeling

Definition 2. A CT-CPN$_S$ is a triple CT-CPN$_S$ = (CPN, P_S, T_S):

(1) CPN is a basic colored Petri nets model;

(2) $P_S = P$, $P_S = P_S^O \cup P_S^E$: P_S^O is the set of *observable places*; P_S^E is the set of *internal places*; $P_S^O \cap P_S^E = \phi$;

(3) $T_S = T$, $T_S = T_S^I \cup T_S^O \cup T_S^E$: T_S^I is the set of *input transitions*; T_S^O is the set of *output transitions*; T_S^E is the set of *internal transitions*; $T_S^I \cap T_S^O = T_S^I \cap T_S^E = T_S^O \cap T_S^E = \phi$;

(4) CT-CPN$_S$ has finite output; that is, $|\text{trace}(M_0)| \leq n$ $(n \in N)$;

(5) CT-CPN$_S$ does not have infinite sequences of internal actions; that is, $\neg\exists M : M \overset{\sigma}{\Rightarrow} M, \sigma \in \text{BE}(T_S^E)^*$.

In the CT-CPN$_S$ modeling, token data in the observable places could present the externally observed data, so an observable place is always the postset of an input transition or an output transition to display what data should be observed after executing those external transitions. The input transition models input actions that accept input data provided by testers and the output transition models output actions that produce visible output observations. Thus, observable places and input/output transitions are used together to explicitly specify external visible behaviors of a certain system. Besides, the internal transitions and internal places could represent internal and unobservable execution of system behaviors. Models must not have loops of internal transitions, which will make system implementations have no response to help us to distinguish this scenario from the deadlock.

4.2. Implementation Modeling. As system implementations are actual physical thing, that is, software, hardware, or a hybrid system, rather than formal objects, a test hypothesis [7] assumes that every system implementation corresponds to an a-priori formal model, but these formal models cannot be explicitly constructed. Therefore, CT-CPN$_I$ is just proposed to formally specify system implementations.

Definition 3. A CT-CPN$_I$ is a triple CT-CPN$_I$ = (CPN, P_I, T_I):

(1) CPN is a basic colored Petri nets model;

(2) $P_I^O = P_S^O$ and $T_I^I = T_S^I$, $T_I^O = T_S^O$ the implementation models and the specification model of the same system having the same observable places and input/output transitions;

(3) $\{\delta\} \subset T_I$: δ is the *suspension transition*, and $M \overset{\delta}{\rightarrow} M$.

Any possible test output, such as real data, deadlock, or quiescence should be managed in CT-CPN$_I$ modeling. In particular, the quiescence represents a scenario where software implementations have no visible outputs because they wait for input action to trigger following executions. Producing quiescence is a kind of special output action, which is modeled as the suspension transition δ. Firing a suspension action indicates that an implementation stays in the same state and needs input data as a trigger to continue executing.

4.3. PN-ioco Relation. In context of CT-CPN modeling, conformance relation should be determined according to specific data in CT-CPN models, so PN-ioco relation is defined.

Definition 4. PN-ioco is a binary relation with $ss \in$ CT-CPN$_S$ and $ii \in$ CT-CPN$_I$:

ii PN-ioco $ss =_{\text{def}} \forall\sigma \in$ SPtrace(M_S): outtoken(M_I fires σ) = outtoken(M_S fires σ).

(1) Consider SPtrace(M_S) $=_{\text{def}} \{\sigma \in (\text{BE}(T_S) \cup \delta)^* \mid M_S \overset{\sigma}{\Rightarrow}\}$; it enumerates all traces of the model ss, including the suspension transitions. M_S and M_I are initial markings, respectively.

(2) Consider outtoken(M) $=_{\text{def}} \{M(P) \mid P \in P_S^O \cup P_I^O\}$; it represents the observable output token data. In the model ss, it records the token data of current observable places under a specific marking, while, in the model ii, it just corresponds to the actual observable output data produced by the system implementations during the test execution.

Guided by the PN-ioco definition, the conformance is determined by comparing token data in the observable places along a specific SPtrace with the actually observed output from the implementation. If the actual observed output data are different from what are prescribed in the ss model, we could conclude with non-conformance decision. The equivalence of two outtoken sets indicates that all prescribed observations should be actually observed in the practical test executions; that is, prescribed functionalities must be completely implemented. Therefore, the implementation that has valid but partial functionalities will not be said to conform to its specification model.

5. Synchronized Simulation Based Test Selection

In Section 3, we mention that, based on CT-CPN modeling and PN-ioco relation, a novel test selection method should be developed with two desired goals. One is to make such test selection with high scalability for dealing with more complicated system models, and the other one is to make all test cases feasible for practical test executions. Accordingly, we develop a test purpose oriented IOCO test selection method to meet these two goals, where test cases are generated through synchronized model simulation between a CT-CPN$_S$ model and a CT-CPN$_{\text{TP}}$ model. When synchronized simulation procedure terminates, a final and feasible test case model is produced and specified as CT-CPN$_{\text{TC}}$.

The intuitive idea of our test selection method is essentially a synchronized traversal between a CT-CPN$_S$ model and a CT-CPN$_{\text{TP}}$ model. It is performed by model simulation execution under given initial markings in these two models. Specifically, each given initial marking in a CT-CPN$_S$ model and a CT-CPN$_{\text{TP}}$ model could conduct a synchronized simulation between these two models once. During following synchronized simulation steps, enabled transitions in both models are capable of firing sequentially in a synchronized way. However, CT-CPN$_{\text{TP}}$ model is responsible for choosing which execution sequence should be cover, and actual test sequences and the data-dependent test oracles are all generated based on CT-CPN$_S$ model. If both models have no further enabled transitions, synchronized model simulation procedure will terminate, and a corresponding test case model is finally generated and specified as a CT-CPN$_{\text{TC}}$ model, while, in context of LTS based test selection in original IOCO testing theory, synchronous product of two LTS model is performed. However, with expansion of model scale, synchronous product operation cannot be executed accurately; thus such LTS based test selection approach fails to support testing large-scale software systems.

In the following subsections, we first propose the formal definitions of test purpose and test case in CT-CPN modeling context. Then, a detailed test selection algorithm is developed with several test generation rules towards different kinds of model transitions. Finally, we adopt a file sharing software system as a representative to demonstrate practical test selection and test execution procedures.

5.1. Test Purpose Modeling. Test purpose is utilized to specify parts of software functional behaviors; for example, in the network based software systems, sending request packets, receiving data packets, or updating local key data structure could be modeled as a test purpose each. In order to promote the pertinence of testing large-scale software systems, test purpose models should participate in test generation process, since it can constrain the scope of test generation; that is, it just selects finite and indispensable test cases to only test expected partial software behaviors under constraint of testing execution time and cost. Obviously, a CT-CPN$_{\mathrm{TP}}$ model is essentially constructed from a corresponding CT-CPN$_S$ model, because it just specifies parts of software behaviors which need to be tested. Therefore, we propose the formal definition of CT-CPN$_{\mathrm{TP}}$ model based on the CT-CPN$_S$ definition.

Definition 5. A CT-CPN$_{\mathrm{TP}}$ is a triple (CT-CPN$_S$, P_{TP}, Σ_{TP}):

(1) CT-CPN$_S$ is a CT-CPN$_S$ model;

(2) $P_{\mathrm{TP}} = P_S \cup \{ph\}$, $\{ph\}$ is the *coverage verdict place*, and only *covered* token can appear in this place;

(3) $\Sigma_{\mathrm{TP}} = \Sigma_S \cup \{covered\}$.

In CT-CPN$_{\mathrm{TP}}$ model, *ph* is always the postset of an output transition and used as the termination place in that model. If a *covered* token appears in *ph* place, it indicates that, based on a given initial marking, a generated test sequence just corresponds to an execution path of behaviors specified in the test purpose model. Thus, we need to add some necessary guard functions towards input or output transitions in the test purpose model to guarantee such behavior path be executed as expected. It is well demonstrated in Section 5.4 through the practical test selection to a file sharing software system.

5.2. Test Case Modeling. Several modeling constraints should be fulfilled in CT-CPN$_{\mathrm{TC}}$ modeling. CT-CPN$_{\mathrm{TC}}$ models should have only one input transition enabled at each step. Besides, they should be deterministic and every feasible trace should have finite length; otherwise, the test execution based on this model cannot terminate in finite steps with the definite testing results.

Definition 6. A CT-CPN$_{\mathrm{TC}}$ is a triple (CPN, P_{TC}, T_{TC}), where

(1) CPN is a basic colored Petri nets model;

(2) $P_{\mathrm{TC}} = P$, $P_{\mathrm{TC}} = P^I_{\mathrm{TC}} \cup P^O_{\mathrm{TC}} \cup P^{\mathrm{TO}}_{\mathrm{TC}} \cup P^V_{\mathrm{TC}}$: P^I_{TC} is the set of *input places*; P^O_{TC} is the set of *observable places*; $P^{\mathrm{TO}}_{\mathrm{TC}}$ is the set of *test oracle places*; P^V_{TC} is the set of *test verdict places*; each pair of them had no intersections;

(3) $T_{\mathrm{TC}} = T$, $T_{\mathrm{TC}} = T^I_{\mathrm{TC}} \cup T^O_{\mathrm{TC}} \cup T^\delta_{\mathrm{TC}} \cup T^V_{\mathrm{TC}}$: T^I_{TC} is the set of *input transitions*; T^O_{TC} is the set of *output transitions*; T^δ_{TC} is the set of *suspension transitions*; T^V_{TC} is the set of *test verdict transitions*; each pair of them had no intersections either;

(4) CT-CPN$_{\mathrm{TC}}$ has finite behavior and is deterministic;

(5) CT-CPN$_{\mathrm{TC}}$ has at most one input transition enabled in each step; that is, $\neg\exists M : M \xrightarrow{(t_1,b_1)} \wedge M \xrightarrow{(t_2,b_2)}$, $t_1 \in T^I_{TC} \wedge t_2 \in T^I_{\mathrm{TC}}$.

It should be noted that, in the test verdict places, only three kinds of token data can appear, that is, *pass/fail/covered* tokens. A *pass* token indicates that current test execution step is successfully passed; a *fail* token indicates an implementation fault with current test execution step and results in the nonconformance decision; a *covered* token indicates that current test execution step covers that behaviors specified in test purpose model.

CT-CPN$_{\mathrm{TC}}$ models could facilitate the actual test execution for their better feasibility and readability, because they not only prescribe the test sequences from the data-dependent control flow of the system behaviors but also provide necessary and definite test oracles for determining the conformance relation.

5.3. Test Selection via Synchronized Model Simulation. To develop a test purpose model oriented IOCO test selection method, we need considering simulation paths in CT-CPN$_S$ model and CT-CPN$_{\mathrm{TP}}$ model at the same time. The reason is that the simulation in the CT-CPN$_S$ model reflects actual execution paths of a software system, including real input and output data, which is the basis of test case generation. However, the simulation in the CT-CPN$_{\mathrm{TP}}$ model conducts to select expected execution paths among all enabled paths. Thus, test generation scope is well constrained into those software behaviors we want to test does not consider other behaviors which are not specified in test purpose models. To accomplish such goal, the CT-CPN$_S$ model and the CT-CPN$_{\mathrm{TP}}$ model should simulate in a synchronized way. Specifically, under guidance of PN-ioco relation, given an initial marking M_S in CT-CPN$_S$ model and an initial marking M_{TP} in CT-CPN$_{\mathrm{TP}}$ model, the IOCO test selection method selects feasible and expected test sequences from the set of SPtrace($M_S \| M_{\mathrm{TP}}$) in order to cover behaviors specified in test purpose models; that is, corresponding enabled transitions in these two models are fired in a synchronized way according to different test generation rules towards different kinds of transitions. Meanwhile, the IOCO test selection method also presents explicit way to decide the conformance relation via test output and test oracle and indicate whether the test purpose is covered. Finally, if no further transitions could fire, that is, a termination marking is reached, the synchronized simulation procedure will terminate. If this final marking represents a valid termination of test purpose directed system behavioral execution, a final CT-CPN$_{\mathrm{TC}}$ test case model covering specific test purpose is generated. Otherwise, if this final marking happens to stand for an invalid deadlock scenario, we need to improve the accuracy of both models and perform the synchronized simulation procedure again.

Rule 1 (synchronized firing rule to input transition). $\exists\ t \in T^I_S$ and $t \in T^I_{\mathrm{TP}}$, $(t, b_S) \in \mathrm{BE}_S$ and $(t, b_{\mathrm{TP}}) \in \mathrm{BE}_{\mathrm{TP}}$ are all enabled: generating $t \in T^I_{\mathrm{TC}}$ and $\forall p \in \{p \in P^I_S \mid p \in \mathrm{pre}(t)\} : p \in P^I_{\mathrm{TC}}$, keeping token data in $M(p)$, and generating a new internal place $p' \in P^E_{\mathrm{TC}} \mid p' = \mathrm{post}(t)$.

Firing (t, b_S) and (t, b_{TP}), that is, $M(p') = \{M \mid M(p) \xrightarrow{(t,b_S)}\}$, to accomplish a synchronized firing of input transitions.

Rule 2 (synchronized firing rule to internal transition). $\exists\, t \in T^E_S$ and $t \in T^E_{\mathrm{TP}}$, $(t, b_S) \in \mathrm{BE}_S$ and $(t, b_{\mathrm{TP}}) \in \mathrm{BE}_{\mathrm{TP}}$ are all enabled, if these two internal transitions could be fired to produce the same token data, then just fire them to accomplish a synchronized firing of internal transitions. Otherwise, fire (t, b_S) time after time until $t \in T^E_{\mathrm{TP}}$; that is, certain internal transition appearing only in the CT-CPN$_S$ model should be firstly fired several times before reaching the state that synchronized internal transition appearing in both CT-CPN$_S$ model and CT-CPN$_{\mathrm{TP}}$ model is able to execute. However, such internal behaviors do not need to be handled by test case models.

Rule 3 (synchronized firing rule to output transition). $\exists\, t \in T^O_S$ and $t \in T^O_{\mathrm{TP}}$, $(t, b_S) \in \mathrm{BE}_S$ and $(t, b_{\mathrm{TP}}) \in \mathrm{BE}_{\mathrm{TP}}$ are all enabled: generating $t \in T^O_{\mathrm{TC}}$ and $\forall p \in \{p \in P^O_S \mid p \in \mathrm{post}(t)\} : p \in P^O_{\mathrm{TC}}$.

If $\exists p \in \{p \in P^E_S \mid p \in \mathrm{post}(t)\}$, we generate a new internal place $p' \in P^E_{\mathrm{TC}} \mid p' = \mathrm{post}(t)$.

Firing (t, b_S) and (t, b_{TP}) to accomplish a synchronized firing of output transitions and constructing a test verdict unit for every $p \in P^O_{\mathrm{TC}}$ are as follows:

(i) generating $q \in P^{\mathrm{TO}}_{\mathrm{TC}}$, $t_v \in T^V_{\mathrm{TC}}$, $p_v \in P^V_{\mathrm{TC}}$: $\{\mathrm{pre}(t_v) = p \cup q\}$ and $\{\mathrm{post}(t_v) = p_v\}$; $M(q) = M(p)$, where $M(q)$ records the test oracle data with respect to p;

(ii) generating $a \in (t_v, p_v)$, $r = \mathrm{pre}(t_v)$ $(r \in P^O_{\mathrm{TC}})$, and, without loss of generality, $E(a)$ could be specified as follows: if $M(r) = M(q)$, then $1'$*pass* + $1'$*covered* else $1'$*fail*; that is, if observed test output in $M(r)$ is the same as the test oracle data in $M(q)$, and test purpose is definitely covered, a *pass* token and a *covered* token are both generated into the test verdict place p_v;

(iii) connecting $t \in T^O_{\mathrm{TC}}$ with newly generated internal place or existing observable place, that is, $r \in P^E_{\mathrm{TC}} \cup P^O_{\mathrm{TC}} \mid r = \mathrm{pre}(t)$ to keep the connectivity of current test case model.

Rule 4 (adding suspension transition). Consider $\exists p \in P^I_{TC}$, if quiescence is allowed, adding a suspension transition with p; that is, $t \in T^\delta_{\mathrm{TC}} : p = \mathrm{pre}(t) \land p = \mathrm{post}(t)$.

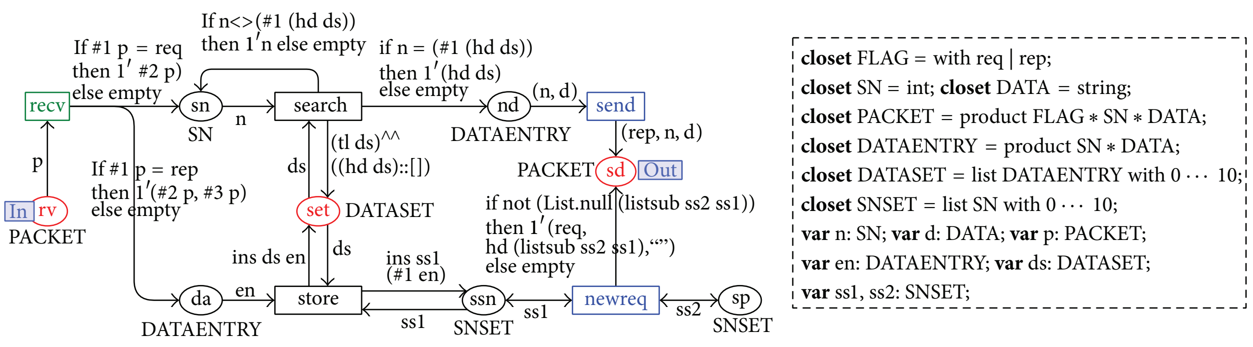
Given sets of initial markings M_S and M_{TP}, through applying a suitable rule of aforesaid four test generation rules step by step, CT-CPN$_{\mathrm{TC}}$ models covering specific test purpose are generated for testing corresponding software behaviors. Besides, it is guaranteed that the scenario never exists where transitions in CT-CPN$_{\mathrm{TP}}$ model are enabled but related transitions in CT-CPN$_S$ model are not enabled, because a CT-CPN$_{\mathrm{TP}}$ model is constructed from the corresponding CT-CPN$_S$ model. Furthermore, based on above test selection method, each SPtrace has finite length, so the test selection algorithm is terminated in finite steps with generating a CT-CPN$_{\mathrm{TP}}$ model that has finite behaviors for practical test executions.

Two more aspects should be noted. First, CT-CPN models are not modified with new kinds of model elements, and modeling constraints are just used to avoid generating infeasible traces for testing scenarios. So, the semantic rules defined in [10] are all kept in CT-CPN models; that is, we still use its original enabling rules and occurrence rules to generate CT-CPN$_{\mathrm{TC}}$ test case models. Second, this test selection approach can be applied into the hierarchical CPN model without modification. The reason comes from two aspects: first, a hierarchical CPN model could definitely be unfolded to a behavioral equivalent nonhierarchical CPN model, and, then, test case selection process just utilizes ordinary simulation techniques which could be applied into hierarchical or nonhierarchical CPN models without differences. Thus, our test selection method has better scalability to deal with most of actual software behavior models.

We could compare the computation cost in test purpose oriented test selection methods. In context of LTS, the state space produced by synchronous product of two LTS models tends to grow exponentially, which needs enormous even unpractical computation resources to generate suitable test cases. But, in context of CPN, synchronized model simulation based selection approach is irrespective of their model scale, so it just needs linear computation cost to produce feasible test cases. The effectiveness and practicability of our test selection method are enhanced with better scalability.

5.4. An Example: File Sharing Software System. We now apply the test purpose oriented IOCO test selection method into a file sharing software system to illustrate its feasibility and effectiveness. The CT-CPN$_S$ model is presented in Figure 2, and one example CT-CPN$_{\mathrm{TP}}$ model is presented in Figure 3. Through synchronized simulating of such two models, a CT-CPN$_{\mathrm{TC}}$ model is generated and shown in Figure 4.

In this file sharing software system, peer nodes that have same functionalities could share the same resource file via network. The file is composed of several file segments which are identified as [SN, DATA]. When a peer node receives a file segment request (req), it searches such data segment from its data segment sets (DATASET) according to the segment number (SN) in request packet. If this peer node gets the requested data segment (DATAENTRY), it immediately sends the segment to the requesting peer node. While as a peer node receives a requested file segment, it stores this segment data into data segment sets firstly and goes on requesting new file segments which this peer does not have yet. In its CT-CPN$_S$ model, *rv*/*sd*/*set* are observable places. *recv* is an input transition to specify the receiving of segment request or data. *send*/*newreq* are output transitions to specify sending segment data, storing segment data and requesting new file segment, respectively. For example, If *send* fires, we could observe which packet is sent according to the tokens in *sd*. The rest are internal places and transitions. In particular, the place *sp* stores the total set of segment number beforehand



```
closet FLAG = with req | rep;
closet SN = int; closet DATA = string;
closet PACKET = product FLAG ∗ SN ∗ DATA;
closet DATAENTRY = product SN ∗ DATA;
closet DATASET = list DATAENTRY with 0 ··· 10;
closet SNSET = list SN with 0 ··· 10;
var n: SN; var d: DATA; var p: PACKET;
var en: DATAENTRY; var ds: DATASET;
var ss1, ss2: SNSET;
```

FIGURE 2: The CT-CPN$_S$ model of the file sharing software system.

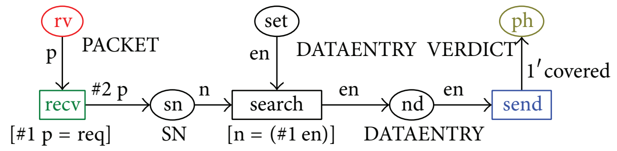


FIGURE 3: CT-CPN$_{TP}$ model of file sharing software system.

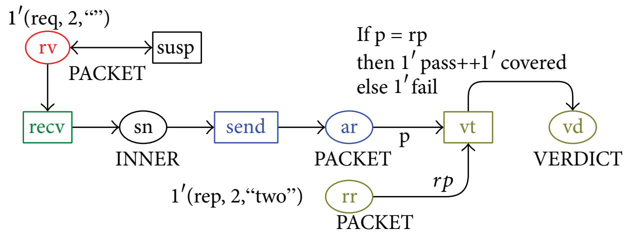


FIGURE 4: CT-CPN$_{TC}$ model of file sharing software system.

to help to choose which segment should be requested. In this way, necessary external behaviors of a system for its conformance testing are modeled accurately. However, in system modeling practice, the selection of observable places should also consider the actual observation points in the actual test execution, such as the points of control and observation [1].

In Figure 3, we present a test purpose CT-CPN$_{TP}$ model as a representative. This test purpose focuses on the function that whether a peer node could get the requested date segment from its data segment sets according to the segment number (constrained by the guard function $[n = (\#1\text{en})]$), when it receives a file segment request (constrained by the guard function $[(\#1p) = \text{req}]$). If a *covered* token appears in *ph* place, it indicates that generated test sequence just covers the behaviors specified in this test purpose model. The CT-CPN$_{TC}$ model in Figure 4 is used to check whether a peer node gets the requested date segment from its data segment sets according to the segment number. The detailed test selection procedure is discussed as follows.

(i) Initial marking is assigned as $M_S(rv) = \{1'(\text{req},2,\text{“”})\}$, $M_S(sp) = \{[1,2,3,4]\}$, $M_S(set) = \{[(1,\text{“one”}), (2,\text{“two”})]\}$; $M_{TP}(rv) = \{1'(\text{req},2,\text{“”})\}$, $M_{TP}(set) = \{[(2,\text{“two”})]\}$.

(ii) Rule 1 is applied to deal with input transition *recv*. This transition could be fired as $(recv, \{p = (\text{req},2,\text{“”})\})$ in a synchronized execution in both models. A new internal place *sn* is added as the postset of *recv* and $M_{TC}(rv) = \{1'(\text{req},2,\text{“”})\}$.

(iii) Rule 2 is applied to deal with internal transition *search*. In the CT-CPN$_S$ model, this transition relates two enabled scenarios, that is, $(search, \{n = 2, \text{ds} = [(1,\text{“one”}), (2,\text{“two”})]\})$ and $(search, \{n = 2, \text{ds} = [(2,\text{“two”}), (1,\text{“one”})]\})$, while, in the CT-CPN$_{TP}$ model, it only relates one enabled scenario, that is, $(search, \{n = 2, \text{en} = (2,\text{“two”})\})$. Thus, we fire *search* in CT-CPN$_S$ model twice, where at the first time it is fired independently, and in the second time it is synchronized fired with that in the CT-CPN$_{TP}$ model. Consider $M_S(nd) = M_{TP}(nd) = \{(2,\text{“two”})\}$.

(iv) Rule 3 is applied to deal with output transition *send*. This transition could be fired as $(send, \{n = 2, d = \text{“two”}\})$ and $(send, \{\text{en} = (2,\text{“two”})\})$ in a synchronized execution in each model and get $M_S(sd) = \{(\text{rep},2,\text{“two”})\}$, $M_{TP}(ph) = \{covered\}$, which indicates that the test purpose covered sequence is executed. The output transition *send* and its postset observable place *ar* are both generated in CT-CPN$_{TC}$ model. Furthermore, its test oracle place *rr*, test verdict transition *vt*, and test verdict place *vd* are all generated as an integrated test verdict unit for validating whether the actual output data is the same with the test oracle data. Consider $M_{TC}(rr) = M_S(sd) = \{(\text{rep},2,\text{“two”})\}$. Finally, we need to relate the output transition *send* with internal place *sn*, that is, $\text{pre}(send) = sn$, to keep the connectivity of generated CT-CPN$_{TC}$ model.

(v) Rule 4 is applied to add a suspension transition towards the input transition *rv*, which allows for waiting to *send* the data packet in a quiescence scenario.

Using above test case model, we perform actual test executions to further illustrate feasibility and effectiveness of our test selection method. We program eight software

Table 1: System implementations and test results.

(a)

Software implementations		
Number	Type	Description
i1	Totally correct	Implementing all functions correctly
i2	Faulty	Error on sending request packet
i3	Faulty	Error on matching segment number
i4	Faulty	Error on storing segment data
i5	Faulty	Error on calculating the new segment number
i6	Faulty	Error on parsing received segment packet
i7	Partially correct	Not supporting segment retrieval
i8	Partially correct	Not supporting segment storage

(b)

Testing results							
i1	i2	i3	i4	i5	i6	i7	i8
Pass	Fail	Fail	Pass	Pass	Fail	Fail	Pass

implementations of the file sharing system with preinjected errors to act as SUT.

In Table 1, software implementations description and testing results are all listed. (1) i1 passes this test case, so it conforms to the specification model in Figure 2. (2) i2, i3, and i6 have fatal errors, respectively, which this test purpose just covers, so they do not pass this test case where *fail* token appears in test case executions. (3) i4 and i5 pass the test case, but the fact is that error behaviors in i4 and i5 are not tested at all by this test case. Test purpose model in Figure 3 does not contain such behaviors; thus definitely the generated test case model does not aim to test these implementation errors. (4) i7 only implements partial functions; that is, it does not support segment retrieval. However, this function is just to be tested by this test case, so i7 does not pass as we do not observe the output segment data packet in actual test execution. Compared with basic IOCO relation definition, i7 does not conform to the specification according to PN-ioco relation, so partially correct implementations are no longer determined with conformance. (5) i8 passes this test case, because the test case does not touch such segment storage functionality.

From the above analysis, we could see that test case models generated using our test selection method are quite feasible for guiding actual test execution intuitively and also effective for finding various implementation faults. According to testing results, conformance relation between a specific implementation and its specification model could be accurately determined.

6. Proof of Soundness and Test Purpose Coverage

The conformance relation between a software implementation *ii* and its specification model *ss* is determined through test executions, specified as *ii* PN-ioco *ss* $\Leftrightarrow$ *ii pass* T_S. If all test cases in completer set T_S are passed, "*ii* PN-ioco *ss*" is consequently determined. However, in practical conformance testing, generating all test cases in the T_S is almost infeasible. Moreover, conformance testing just aims to find nonconformance faults rather than to completely prove the conformance. Thus, a weaker requirement is usually considered; that is, as long as the implementation does not pass one test case, it definitely does not conform to its specification. This weaker requirement corresponds to the left-to-right implication of *ii* PN-ioco *ss* $\Leftrightarrow$ *ii pass* T_S and is referred as the soundness of test case generation approach.

Theorem 7. *Let ss* $\in$ *CT-CPN*$_S$ *be a specification model, and tp* $\in$ *CT-CPN*$_{TP}$ *a test purpose model; let* T_S *be a complete set of test cases that generated from ss and tp with our test selection algorithm, that is, TestSel; let CT-CPN*$_S$ $\times$ *CT-CPN*$_{TP}$ $\rightarrow$ *CT-CPN*$_{TC}$ *be the test case selecting function that satisfies TestSel*$(ss, tp) \subseteq T_S$*; then TestSel is sound for ss with respect to PN-ioco.*

Proof. Supposing $\exists ii \in$ CT-CPN$_I$ with corresponding $ss \in$ CT-CPN$_S$, $\exists tp \in$ CT-CPN$_{TP}$ and $\exists tt \in$ *TestSel(ss,tp)* satisfying *not* (*ii pass tt*) and *ii* PN-ioco ss, then
not (ii pass tt)

$\Rightarrow \exists e \in M(p), p \in P_{TS}^{V}, e \in \{fail\}$; [a *fail* token appears in the test verdict place *p*]

$\Rightarrow \exists r \in P_{TS}^{O} \wedge \exists q \in P_{TS}^{TO}, M(r) \neq M(q)$; [token data in the observable place *r* and its coupled test oracle place *q* are different]

$\Rightarrow \exists \sigma \in$ SPtrace(M_S): outtoken$(M_I$ fires $\sigma) \neq$ outtoken$(M_S$ fires $\sigma)$, where $M(r) \subseteq (M_I$ fires $\sigma)$ and $M(q) \subseteq (M_S$ fires $\sigma)$. [trace σ results in unexpected output observations in the practical test execution].

Obviously, there exists a contradiction with the assumption *ii* PN-ioco *ss*. Therefore, we could conclude that if one test case does not pass, that is, *not* (*ii pass tt*), then, *ii* PN-ioco *ss* does not hold definitely; that is, *TestSel* is sound for *ss* and *tp* with respect to PN-ioco. However, it should be noted that *TestSel* is not empty since SPtrace is always produced as a specific initial marking is assigned in actual *ss* and *tp* models. □

It should be noted that as CT-CPN$_{TP}$ model is constructed from its corresponding CT-CPN$_S$ model, when valid specific initial markings are assigned in actual CT-CPN$_S$ model and CT-CPN$_{TP}$ model, at least one SPtrace exists definitely, so at least one test case model is generated. Given sets of initial markings, several test case models covering specific test purposes are generated consequently. That is, the special case where empty set of *TestSel(ss,tp)* tends to be sound never exists.

Based on the guarantee that test case selection is sound, we need further guarantee that test selection should cover test purposes; that is, any passed test case is definitely testing and only testing those software behaviors which are specified in corresponding test purpose models. The coverage towards test purpose is formally described as *cover-pass* relation as follows:

$\exists ii \in$ CT-CPN$_I$ with related $ss \in$ CT-CPN$_S$, $\exists tp \in$ CT-CPN$_{TP}$, $\exists tt \in T_S$(a complete set of test cases generated from *ss* and *tp*):

ii cover-pass tt $\rightarrow$ *ii exhibit tp:*

ii cover-pass tt $=_{\text{def}}$ *ii pass tt* and $\{covered\} \in M(P_{\text{TC}}^V)$;

ii exhibit tp $=_{\text{def}} \forall\sigma \in$ SPtrace(M_{TP}): outtoken$(M_I$ fires $\sigma) \supseteq$ outtoken$(M_{\text{TP}}$ fires $\sigma)$.

From the angle of converse negative proposition in above relation, we could conclude that if a system implementation does not perform the behaviors which are specified in a test purpose model, it must not pass any test case generated using such test purpose model.

Theorem 8. *Let ss* $\in$ *CT-CPN*$_S$ *be a specification model and tp* $\in$ *CT-CPN*$_{TP}$ *a test purpose model; let* T_S *be a complete set of test cases that generated from ss and tp with our test selection algorithm, that is, TestSel, let CT-CPN*$_S \times$ *CT-CPN*$_{TP} \rightarrow$ *CT-CPN*$_{TC}$ *be the test selecting function that satisfies TestSel(ss, tp)* $\subseteq T_S$*; then TestSel covers behaviors in tp model.*

Proof. $\exists ii \in$ CT-CPN$_I$ with corresponding $ss \in$ CT-CPN$_S$, $\exists tp \in$ CT-CPN$_{\text{TP}}$ and $\exists tt \in$*TestSel(ss,tp)*:

ii cover-pass tt

$\Rightarrow$ *ii pass tt* $\wedge \exists p \in P_{\text{TC}}^V, \{covered\} \in M(p)$

$\Rightarrow \forall p' \in P_{\text{TC}}^V, \{fail\} \notin M(p') \wedge \exists p \in P_{\text{TC}}^V, \{covered\} \in M(p)$ [only *pass* and *covered* token data can appear in the test verdict place]

$\Rightarrow \forall\sigma \in$ SPtrace$(M_S \| M_{\text{TP}})$: outtoken$(M_{\text{TP}}$ fires $\sigma) \subseteq$ outtoken$(M_S$ fires $\sigma)$ and outtoken$(M_S$ fires $\sigma) =$ outtoken$(M_I$ fires $\sigma)$

$\Rightarrow \forall\sigma \in$ SPtrace$(M_S \| M_{\text{TP}})$: outtoken$(M_{\text{TP}}$ fires $\sigma) =$ outtoken$(M_I$ fires $\sigma)$ [trace σ results in expected output observations in the practical test execution].

□

7. Conclusion

To make the best of advantages of the IOCO testing theory and the CPN modeling, we integrate them directly to develop a novel test purpose model oriented IOCO test selection method. Based on conformance testing oriented CPN models for specifying software functional behaviors and specific test purposes, respectively, guided by the CPN based IOCO relation, feasible test cases are generated using synchronized model simulation with the proof of the soundness of test generation and the coverage towards test purposes. Throughout practical test selection and test execution for a file sharing software system as a representative, the feasibility and effectiveness of the preceding test selection method are well elaborated.

Our CPN model based IOCO test selection method has several advantages. First, conformance test cases are generated through synchronized simulation process with actual test input data and test oracles, so they are well feasible for guiding practical testing executions. Second, as synchronized model simulations with two CPN models are irrespective of their model scale, their effectiveness and practicability are enhanced with better scalability. Therefore, our CPN model based IOCO test selection method is promising and competent for validating the correctness of reactive network software systems more efficiently and more effectively.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

This work was supported in part by the National Natural Science Foundation of China (61262017 and 61262082), the Key Project of Chinese Ministry of Education (212025), the Inner Mongolia Science Foundation for Distinguished Young Scholars (2012JQ03), and the Introduction Foundation for High-Level Talents of Inner Mongolia University. The authors wish to thank the anonymous reviewers for their helpful comments in reviewing this paper.

References

[1] International Organization for Standardization, *Information Technology—Open Systems Interconnection—Conformance Testing Methodology and Framework—Part 1: General Concepts*, ISO/IEC 9646-1, International Organization for Standardization, Geneva, Switzerland, 2nd edition, 1994.

[2] R. M. Hierons, K. Bogdanov, J. P. Bowen et al., "Using formal specifications to support testing," *ACM Computing Surveys*, vol. 41, no. 2, article 9, 2009.

[3] S. R. Dalal, A. Jain, N. Karunanithi et al., "Model-based testing in practice," in *Proceedings of the International Conference on Software Engineering (ICSE '99)*, pp. 285–294, Los Angeles, Calif, USA, May 1999.

[4] J. Yan, J. Wang, and H. W. Chen, "Survary of model-based software testing," *Computer Science*, vol. 31, no. 2, pp. 184–187, 2004 (Chinese).

[5] M. Broy, B. Jonsson, J. P. Katoen, M. Leucker, and A. Pretschner, *Model-Based Testing of Reactive Systems*, vol. 3472 of *Lecture Notes in Computer Science*, Springer, Heidelberg, Germany, 2005.

[6] J. Tretmans, "Model based testing with labelled transition systems," in *Formal Methods and Testing*, vol. 4949 of *Lecture Notes in Computer Science*, pp. 1–38, Springer, Heidelberg, Germany, 2008.

[7] J. Tretmans, "Test generation with inputs, outputs and repetitive quiescence," *Software-Concepts and Tools*, vol. 17, no. 3, pp. 103–120, 1996.

[8] J. Tretmans and E. Brinksma, "TorX: automated model based testing," in *Proceedings of the 1st European Conference on Model-Driven Software Engineering (ECMDSE '03)*, pp. 1–13, Nuremberg, Germany, December 2003.

[9] C. Jard and T. Jéron, "TGV: theory, principles and algorithms. A tool for the automatic synthesis of conformance test cases for non-deterministic reactive systems," *International Journal on Software Tools for Technology Transfer*, vol. 7, no. 4, pp. 297–315, 2005.

[10] K. Jensen and L. M. Kristensen, *Coloured Petri Nets: Modelling and Validation of Concurrent Systems*, Springer, Heidelberg, Germany, 2009.

[11] J. Liu, X.-M. Ye, and J. Li, "Colored Petri nets model based conformance test generation," in *Proceedings of the 16th IEEE Symposium on Computers and Communications (ISCC '11)*, pp. 967–970, Corfu, Greece, July 2011.

[12] R. G. Vries and J. Tretmans, "Towards formal test purposes," in *Proceedings of the 1st International Workshop on Formal Approaches to Testing of Software (FATES '01)*, pp. 61–76, Aarhus, Denmark, August 2001.

[13] M. Weiglhofer, G. Fraser, and F. Wotawa, "Using coverage to automate and improve test purpose based testing," *Information and Software Technology*, vol. 51, no. 11, pp. 1601–1617, 2009.

[14] T. Jéron, "Symbolic model-based test selection," *Electronic Notes in Theoretical Computer Science*, vol. 240, no. 7, pp. 167–184, 2009.

[15] H. Watanabe and T. Kudoh, "Test suite generation methods for concurrent systems based on coloured Petri nets," in *Proceedings of the 2nd Asia-Pacific Software Engineering Conference (APSEC '95)*, pp. 242–251, Brisbane, Australia, 1995.

[16] U. Farooq, C. P. Lam, and H. Li, "Towards automated test sequence generation," in *Proceedings of the 19th Australian Software Engineering Conference (ASWEC '08)*, pp. 441–450, Perth, Australia, March 2008.

[17] H. Zhu and X.-D. He, "A methodology of testing high-level Petri nets," *Information and Software Technology*, vol. 44, no. 8, pp. 473–489, 2002.

[18] Y. Ledru, L. du Bousquet, P. Bontron, O. Maury, C. Oriat, and M. L. Potet, "Test purposes: adapting the notion of specification to testing," in *Proceedings of the 16th Annual International Conforence on Automated Software Engineering (ASE '01)*, pp. 127–134, Antwerp, Belgium, November 2001.

[19] K. P. Chan, T. Y. Chen, and D. Towey, "Good random testing," in *Proceedings of the 9th International Conforence on Reliable Software Technology (Ada-Europe '04)*, pp. 200–212, Palma de Mallorea, Spain, 2004.

[20] C. Pachecol, S. K. Lahiri, M. D. Ernst, and T. Ball, "Feedback-directed random test generation," in *Proceedings of the 29th International Conforence on Software Engineering (ICSE '07)*, pp. 75–84, Minneapolis, Minn, USA, May 2007.

[21] J. Fernandez, L. Mounier, and C. Pachon, "Property oriented test case generation," in *Proceedings of the 3rd International Workshop on Formal Approaches to Testing of Software (FATES '03)*, pp. 147–163, Montreal, Canada, October 2003.

[22] G. Fraser, F. Wotawa, and P. E. Ammann, "Testing with model checkers: a survey," *Software Testing Verification and Reliability*, vol. 19, no. 3, pp. 215–261, 2009.

Hindawi Publishing Corporation
Journal of Applied Mathematics
Volume 2014, Article ID 920123, 13 pages
http://dx.doi.org/10.1155/2014/920123

Research Article

A Bio-Inspired QoS-Oriented Handover Model in Heterogeneous Wireless Networks

Daxin Tian,[1,2] Jianshan Zhou,[1,2] Honggang Qi,[3] Yingrong Lu,[1,2] Yunpeng Wang,[1,2] Jian Wang,[1,2] and Anping He[4]

[1] *Beijing Key Laboratory for Cooperative Vehicle Infrastructure Systems & Safety Control, School of Transportation Science and Engineering, Beihang University, Beijing 100191, China*
[2] *Jiangsu Province Collaborative Innovation Center of Modern Urban Traffic Technologies, Beijing 100191, China*
[3] *School of Computer and Control Engineering, University of Chinese Academy of Science, Beijing 100049, China*
[4] *Guangxi Key Laboratory of Hybrid Computation and IC Design Analysis, Guangxi University for Nationalities, Nanning 530006, China*

Correspondence should be addressed to Honggang Qi; hgqi@ucas.ac.cn and Anping He; hapetis@gmail.com

Received 11 February 2014; Accepted 20 March 2014; Published 14 April 2014

Academic Editor: Xiaoyu Song

We propose a bio-inspired model for making handover decision in heterogeneous wireless networks. It is based on an extended attractor selection model, which is biologically inspired by the self-adaptability and robustness of cellular response to the changes in dynamic environments. The goal of the proposed model is to guarantee multiple terminals' satisfaction by meeting the QoS requirements of those terminals' applications, and this model also attempts to ensure the fairness of network resources allocation, in the meanwhile, to enable the QoS-oriented handover decision adaptive to dynamic wireless environments. Some numerical simulations are preformed to validate our proposed bio-inspired model in terms of adaptive attractor selection in different noisy environments. And the results of some other simulations prove that the proposed handover scheme can adapt terminals' network selection to the varying wireless environment and benefits the QoS of multiple terminal applications simultaneously and automatically. Furthermore, the comparative analysis also shows that the bio-inspired model outperforms the utility function based handover decision scheme in terms of ensuring a better QoS satisfaction and a better fairness of network resources allocation in dynamic heterogeneous wireless networks.

1. Introduction

Recently, with the highly developed wireless communication technologies such as the wireless LANs (WLAN IEEE 802.11a/b/g/n/p standards), WiMAX (IEEE 802.16a/e standards), the third and fourth generation cellular wireless (3G or 4G), and satellite communications, substantial significant infrastructure of these wireless networks has been deployed to support the dramatically increasing demand for mobile terminals' access to network services anywhere and anytime. However, no single wireless access technology can be efficient to guarantee various users' demands for reliable connection and quality of service (QoS) over all situations. Consequently, the next generation wireless communication system is evolving, which depends on those heterogeneous wireless networks. In the nowadays heterogeneous wireless networks, multiple wireless access technologies as well as multiple radios have to interwork and to be used in a cooperative manner to realize the "always best connected" (ABC) concept in terms of high level of QoS satisfaction and fairness resources allocation [1].

Handover decision is one of the most important issues (including handover management, resource allocation, and mobility support) related to the heterogeneous wireless networks and should be efficiently addressed for the realization of the envisioned next generation communication system [2]. However, there exist some significant challenges in developing the essential functional components and in designing the corresponding algorithms for handover decision such as impracticality of centralized control, dynamic nature, resources constraint, and heterogeneity in the heterogeneous wireless environment [3].

On the other hand, the bio-inspired paradigm provides a novel approach to design a new powerful solution for many engineering problems [4, 5]. Similar challenges met by the heterogeneous wireless communication system arising from dynamic nature, system complexity, heterogeneous architectures, and absence of centralized control have been well addressed by the biological system [6]. Many biological mechanisms such as the adaptability to environmental changes, inherent robustness to external perturbation, and self-optimization are appealing to be introduced in the handover decision solution to deal with those aforementioned significant challenges. As some interdisciplinary studies have argued, many biological mechanisms resulting from the evolution of nature over millions of years always go far away beyond the traditional technologies so that they are promising to be used to settle some complex engineering problems [7–9].

The attractor selection is one type of bio-inspired mechanism that induces cellular gene expression to adaptively respond to the dynamically changing environment. Its related model, that is, attractor selection model [10], has attracted much attention and has been extended to be implemented in many engineering domains consequently. For example, it has been applied in the robust robot control [11], the error-tolerant wireless sensor networks control [12], the adaptive virtual network topology control [13], the adaptive routing protocol in mobile ad hoc networks or overlay networks [14], and the resources allocation among multiple users and multiple applications in the heterogeneous wireless environment [15].

Motivated by the attractor selection of cellular gene network, we adopt this bio-inspired mechanism for modeling the vertical handover decision in heterogeneous wireless networks. The goal of the paper is to deal with the varying heterogeneous wireless environmental conditions, at the same time, to guarantee users' satisfaction level, and to ensure the fairness of network resources allocation among multiple mobile terminals. We extend the basic attractor selection model which has been proposed in [10] to a novel one with a higher dimension for multiple attributes decision making. The upper and lower bounds of QoS requirements of multiple applications are combined with the dynamic wireless network conditions including bandwidth, end-to-end delay, and packet loss ratio to formulate a function for evaluating the terminal's QoS satisfaction. This utility function is used to assist the bio-inspired model in performing the attractor selection mechanism. And the handover decision induced by the attractor selection mechanism allows us to capture the dynamic nature of the heterogeneous wireless environment and to evaluate the goodness of accessing wireless networks, so as to enable the handover decision adaptive to the wireless environmental changes.

The remainder of this paper is organized as follows. Section 2 formulates the handover decision problem to be solved and introduces the basic attractor selection model so as to extend it for making handover decision. The QoS-oriented handover decision framework and the detailed scheme based on the proposed bio-inspired handover model are described in Section 3. Section 4 demonstrates some comparative simulation results and gives the analysis of our handover decision scheme. Finally, Section 5 concludes this work.

2. System Model

In this section, the core problem to be solved in the handover decision is described firstly, and then the basic attractor selection model as well as its corresponding biomechanism is presented. Following the mathematical form of the basic attractor selection model, we extend it to a novel form and apply this extended model for multiple attributes decision making.

2.1. Problem Formulation. We assume that each mobile terminal moving in a given heterogeneous wireless environment is equipped with a multimode communication device. These terminals with multiple wireless interfaces are able to access different wireless networks simultaneously. Namely, in this assumption, a mobile user is allowed to assign different wireless links to its different applications that are running in the terminal device. We consider a heterogeneous wireless environment composed of a network set of multiple heterogeneous wireless networks. We denote this network set as $\text{NetSet} = \{\text{net}_1, \text{net}_2, \ldots, \text{net}_M\}$. The parameter M, here, denotes the total number of those considered wireless networks. And then we consider that there are totally Q vehicular terminals moving in this given heterogeneous wireless environment. All of these mobile terminals compose a set that is denoted by $\text{User} = \{u_1, u_2, \ldots, u_Q\}$, and each one $u_k \in \text{User}$ has a certain number of applications running in its terminal device. For instance, we denote those applications simultaneously running in u_k as a set $S_{u_k} = \{s_1, s_2, \ldots, s_N\}$. Thus, each u_k is required to make a decision to select the most suitable wireless network for each of its applications. Each application $s_i \in S_{u_k}$ may connect to the same network or may use different wireless links. Each u_k performs the handover decision process during every discrete period Δt. When the network selection is done, the wireless interface of each application in u_k is switched from the previous one to the new determined network.

2.2. Basic Attractor Selection Model. The basic attractor selection model is inspired from cell biology. It is used to describe the adaptive response of the gene expression in an *Escherichia coli* (*E. coli*) cell to the changes in its available nutrients, especially when there is no enough molecular machinery for signal transduction from the environment to the DNA expression [10]. The basic mathematical model of the attractor selection can be expressed by two nonlinear differential equations with stochastic noise as follows:

$$\begin{aligned} \frac{dm_1}{dt} &= \frac{S(A)}{1+(m_2)^2} - D(A) \times m_1 + \eta_1, \\ \frac{dm_2}{dt} &= \frac{S(A)}{1+(m_1)^2} - D(A) \times m_2 + \eta_2, \end{aligned} \tag{1}$$

where m_1 and m_2 represent two different mRNA concentrations, respectively, corresponding to two different nutrients.

$S(A)$ and $D(A)$ are, respectively, two different rate coefficients of nutrient synthesis and degradation. They are defined as the monotonously increasing function of cellular activity that is represented by the parameter A. In [10], $S(A) = 6A/(A+2)$ and $D(A) = A$. η_1 and η_2 represent independent Gaussian white noise which is inherent in cellular gene expression.

From the viewpoint of dynamics system, the variable pair $[m_1, m_2]^T$ can be treated as a state of cellular metabolic phenotype and (1) essentially represents a cell dynamics system. An attractor is a stable state of the dynamics system to which the phase space trajectory of the system will converge, no matter what the initial conditions are. In fact, fluctuation inherently exists in the actual biological system, so the gene expression or other behaviors of a biological entity are not purely deterministic. Even though the state of the cellular dynamics system is perturbed by stochastic noise arising from the external environmental fluctuations, the system is able to gradually come to stability over time and finally stays at a new growth rate as well as at a well living state. The adaption of a cell to changes in its survival environment is analogous to the attractor selection of the dynamics system given in (1), in which the system will select and switch to a new stable and suitable state when the environmental conditions have been changed or perturbed to be unsuitable for previous state.

In addition, the cellular activity A is an important parameter that lumps the fitness of the environmental conditions for the cell's survival to a single real value and it ranges from 0 to 1. This parameter is used to comprehensively reflect the information of the cell's external environment, control the influence of the noise on the behavior of the dynamics system, and capture the phenotypic consequence that enables the cellular adaptation.

2.3. Extended Attractor Selection Model. Following the basic attractor selection model, we would like to introduce the appealing bio-inspired attractor selection mechanism to decision making under varying conditions in terms of improving the robustness and adaptability of the decision solution. In order to select the most appropriate wireless access network $\text{net}_j \in \text{NetSet}$ for any one application $s_i \in S_{u_k}$ that is running in the terminal device of a user $u_k \in \text{User}$, we firstly define a decision vector as $\mathbf{X}^{s_i}(t) = [x^{s_i}_{\text{net}_j}(t)]_{\text{net}_j \in \text{NetSet}} = [x^{s_i}_{\text{net}_1}(t), x^{s_i}_{\text{net}_2}(t), \ldots, x^{s_i}_{\text{net}_M}(t)]^T$, for each application $s_i \in S_{u_k}$. Each state value $x^{s_i}_{\text{net}_j}(t)$ in this decision vector refers to the score or the goodness of the network net_j relevant to the application s_i at time t. Therefore, any $u_k \in \text{User}$ should maintain a set of $|S_{u_k}|$ decision vectors, since it has $|S_{u_k}|$ applications (it should be noted that the notation $|S_{u_k}|$ represents the number of elements in the set S_{u_k}). Then, we use these decision vectors to construct a matrix as follows:

$$[\mathbf{X}^{s_1}(t), \mathbf{X}^{s_2}(t), \ldots, \mathbf{X}^{s_N}(t)] = \begin{bmatrix} x^{s_1}_{\text{net}_1}(t) & x^{s_2}_{\text{net}_1}(t) & \cdots & x^{s_N}_{\text{net}_1}(t) \\ x^{s_1}_{\text{net}_2}(t) & x^{s_2}_{\text{net}_2}(t) & \cdots & x^{s_N}_{\text{net}_2}(t) \\ \vdots & \vdots & \ddots & \vdots \\ x^{s_1}_{\text{net}_M}(t) & x^{s_2}_{\text{net}_M}(t) & \cdots & x^{s_N}_{\text{net}_M}(t) \end{bmatrix}. \tag{2}$$

The matrix shown in (2) is called a "decision matrix" or a "score matrix," whose rows index different wireless networks and columns index different applications. Each component value $x^{s_i}_{\text{net}_j}(t)$ in this decision matrix indicates the proportion of selecting the wireless network net_j as the target network for supporting the application s_i at time t. Furthermore, once the decision matrix can be obtained, we can use (3) to determine the target network for the application s_i:

$$\text{net}^*(s_i) = \underset{\text{net}_j \in \text{NetSet}}{\text{argmax}} \left\{ x^{s_i}_{\text{net}_j}(t) \right\}. \tag{3}$$

Aiming to update the state of the decision matrix, we here extend the basic attractor selection model to the new form with a higher dimension. We propose the novel extended model as follows:

$$\frac{dx^{s_i}_{\text{net}_j}(t)}{dt} = \frac{\text{syn}(\alpha)}{1 + \left[x^{s_i}_{\text{net}^*(s_i)}(t) - x^{s_i}_{\text{net}_j}(t) \right]^2} - \deg(\alpha) \times x^{s_i}_{\text{net}_j}(t) + \eta^{s_i}_{\text{net}_j}(\mu, \sigma), \tag{4}$$

where $s_i \in S_{u_k}$, $\text{net}_j \in \text{NetSet}$ and $x^{s_i}_{\text{net}^*(s_i)}(t)$ is the maximum state value in the decision vector $\mathbf{X}^{s_i}(t)$ corresponding to the application s_i.$\eta^{s_i}_{\text{net}_j}(\mu, \sigma)$ denotes the Gaussian white noise whose mean value is μ and whose standard deviation is σ. According to the basic attractor selection model, $\text{syn}(\alpha)$ and $\deg(\alpha)$ should be designed as the monotonously increasing functions of the activity α. Similar to [14], we adopt the polynomial form to formulate $\text{syn}(\alpha)$ and directly set $\deg(\alpha)$ identical to α as follows:

$$\begin{gathered} \text{syn}(\alpha) = \alpha \times (\beta \times \alpha^n + m), \\ \deg(\alpha) = \alpha, \end{gathered} \tag{5}$$

where β and m are both the positive real values and n is a positive integer. On the basis of (4), we can use the model defined in (5) to dynamically and self-adaptively update the decision matrix defined in (2). Thus, based on the decision matrix, the handover decision can be made in terms of guaranteeing the terminal QoS satisfaction and self-adaptability.

2.4. Model Validation and Discussion. To analyze the proposed model given by (4), we split the stochastic nonlinear differential equation into two parts, that is, the deterministic term and the stochastic term. We define the notation $\Phi^{s_i}_{\text{net}_j}(\alpha, t)$ to represent the deterministic term

$$\Phi^{s_i}_{\text{net}_j}(\alpha, t) = \frac{\text{syn}(\alpha)}{1 + \left[x^{s_i}_{\text{net}^*(s_i)}(t) - x^{s_i}_{\text{net}_j}(t) \right]^2} - \deg(\alpha) \times x^{s_i}_{\text{net}_j}(t). \tag{6}$$

$\Phi^{s_i}_{\text{net}_j}(\alpha, t)$ is a complex multivariate function of the deterministic parameters α and t. Following the notation above, we then reshape the extended attractor selection model defined in (4) into a more simple formulation which is composed of the deterministic term $\Phi^{s_i}_{\text{net}_j}(\alpha, t)$ and the stochastic term $\eta^{s_i}_{\text{net}_j}(\mu, \sigma)$:

$$\frac{dx^{s_i}_{\text{net}_j}(t)}{dt} = \Phi^{s_i}_{\text{net}_j}(\alpha, t) + \eta^{s_i}_{\text{net}_j}(\mu, \sigma). \tag{7}$$

From (7), it can be obviously observed that if the value of the activity α decreases due to some changes in the external conditions of this dynamics system, the magnitude of the deterministic term $\Phi^{s_i}_{\text{net}_j}(\alpha, t)$ will become smaller and it may decreasingly approach the magnitude of the stochastic term $\eta^{s_i}_{\text{net}_j}(\mu, \sigma)$, which means that $|\Phi^{s_i}_{\text{net}_j}(\alpha, t)| \approx |\eta^{s_i}_{\text{net}_j}(\mu, \sigma)|$. In this situation, (7) tells us that the influence of the random noise becomes relatively enhanced and the behavior of the dynamics system is expected to be mostly dominated by the randomness of $\eta^{s_i}_{\text{net}_j}(\mu, \sigma)$. On the other hand, when the activity α increases to make the magnitude of the deterministic term $\Phi^{s_i}_{\text{net}_j}(\alpha, t)$ much larger than that of the stochastic term $\eta^{s_i}_{\text{net}_j}(\mu, \sigma)$, the deterministic term will govern this system so that its phase trajectory can asymptotically approach a more stable state against the fluctuation resulting from stochastic noise. At this point, the process of tending to a stable state can be analogous to the adaptive attractor selection in the gene expression of cells, since the dynamics of gene expression switches between different patterns, which is as well influenced by a combination of the deterministic and the stochastic behavior of cellular system.

Additionally, when the stochastic term $\eta^{s_i}_{\text{net}_j}(\mu, \sigma)$ is assumed to be zero, we can easily obtain the deterministic maximum value of the system state variable by setting $\Phi^{s_i}_{\text{net}_j}(\alpha, t) = 0$ and $\text{net}_j = \text{net}^*(s_i)$ as follows:

$$\begin{aligned}\Phi^{s_i}_{\text{net}_j}(\alpha, t) &= \frac{\text{syn}(\alpha)}{1 + \left[x^{s_i}_{\text{net}^*(s_i)}(t) - x^{s_i}_{\text{net}_j}(t)\right]^2} - \deg(\alpha) \times x^{s_i}_{\text{net}_j}(t) \\ &= \text{syn}(\alpha) - \deg(\alpha)\, x^{s_i}_{\text{net}^*(s_i)}(t) = 0.\end{aligned} \tag{8}$$

Therefore, the deterministic state of maximum value at time t derived from (8) is

$$x^{s_i}_{\text{net}^*(s_i)}(t) = \frac{\text{syn}(\alpha)}{\deg(\alpha)}. \tag{9}$$

In the simple case, with considering the specific formulations of $\text{syn}(\alpha)$ and $\deg(\alpha)$ given by (5), we can get the closed-form expression for $x^{s_i}_{\text{net}^*(s_i)}(t)$ as

$$x^{s_i}_{\max} = \frac{\alpha(\beta \times \alpha^n + m)}{\alpha} = \beta \times \alpha^n + m, \tag{10}$$

where we use the notation $x^{s_i}_{\max}$ to represent $x^{s_i}_{\text{net}^*(s_i)}(t)$ for simplicity.

As mentioned before, we consider that these parameters β, n, and m are positive real values. Thus, from (10), $x^{s_i}_{\max}$ is an increasing function of the activity α. This means that when α is increased, the maximum system state is also increased along with this increasing α. From the biological perspective, if cells successfully express their genes that are able to make them well survive and optimally grow in an uncertain and highly dynamic environment, their activity is consequently expected to approach the best level. It can be said that those cells select the adaptive attractor that corresponds to their suitable gene expression pattern. From this point, the level of the maximum system state given by (10) can capture the fitness degree of the dynamics system in the dynamic environment once the activity α is correlated with the varying environmental conditions. This also explains that it is appropriate to adopt the form of increasing function to represent $\text{syn}(\alpha)$ and $\deg(\alpha)$.

Next, we come to consider the influence of random noise on the dynamics system. In fact, it is impossible to achieve the aforementioned selection of different attractors when only considering the deterministic term $\Phi^{s_i}_{\text{net}_j}(\alpha, t)$ in the dynamics system defined in (7). The biological system is always evolving along with inherent randomness. As [16, 17] have discussed, the stochastic fluctuation in cell systems is one of the most significant factors to drive the process of gene expression switching between attractor states. Some numerical simulations are done to validate how the deterministic and the stochastic terms affect the behavior of the dynamics system represented by the extended attractor selection model.

Following the basic attractor selection model defined in (1), we adopt the assumption of Gaussian white noise for the stochastic term $\eta^{s_i}_{\text{net}_j}(\mu, \sigma)$, and then without loss of generality we set $\mu = 0$ in the following experiments. In the simulation, the standard deviation σ is, respectively, set to be 0.5, 1, and 2.5, and the parameters β, n, and m are fixed at 5, 3, and 5, respectively. These simulations allow us to perform comparative analysis and investigate the properties of the bio-inspired model in the dynamic environment of different stochastic perturbation magnitudes.

Furthermore, because our goal here is to investigate the properties of the proposed model in terms of different random noises, without loss of generality we consider the specific form of the model with only three state variables, respectively, denoted by x_1, x_2, and x_3:

$$\frac{dx_i}{dt} = \frac{\text{syn}(\alpha)}{1 + (x_{\max} - x_i)^2} - \deg(\alpha) \times x_i + \eta_i(0, \sigma), \tag{11}$$

where $i = 1, 2, 3$ and $x_{\max} = \max_{i=1,2,3}\{x_i\}$. Also, it is worth pointing out that the state variables are all restricted in the positive real number domain; that is, $x_i \geq 0$, for all $i = 1, 2$, and 3; since these state variables have actual meaning when they are related to the level of cells gene expression, we will reset them to 0 when any one state variable changes to be lower than zero.

Now, we set the simulation time from 0 seconds to 3600 seconds. And then we employ a changing environment in the simulations where a square wave with period 900 seconds is

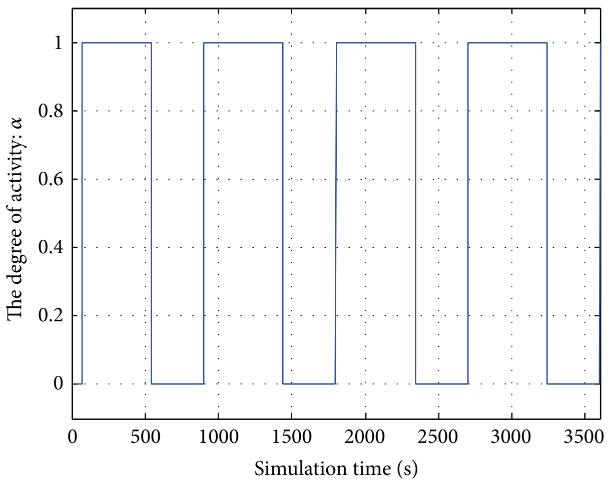


Figure 1: The varying activity α in simulations.

adopted to simulate the time series for the varying activity α. The maximum and the minimum peaks of the square wave are set to be 1 and 0, respectively, and its duty cycle (i.e., the percent of the period in which the signal is larger than 0) is set to be 60%. The simulated varying activity α is given in Figure 1.

In Figure 2, the subgraphs (a), (b), and (c), respectively, show the variation of the dynamics system. In the first case, with the standard deviation of the random noise $\sigma = 0.5$, the dynamics system composed of x_i ($i = 1, 2, 3$) stably stays at the attractor whose maximum state is x_2 and whose lower states are x_1 and x_3. It is because the magnitude of the stochastic term in the model is very small, while the deterministic term always dominates the behavior of the dynamics system even when the activity α decreases to 0 (see Figure 1). This means that the system is trapped into one attractor and will not switch to another. When the magnitude of the stochastic term increases (namely, the standard deviation of the random noise is set to be larger as shown in the cases of (b) and (c)), the state of the dynamics system fluctuates more fiercely with the low value of the activity α. However, when the activity α jumps from the minimum to the maximum peak, the dynamics system always evolves to a stable state, that is, one attractor. For instance, in the subgraph (c), the system stays at the attractor whose maximum state value is x_1 and whose lower states are x_2 and x_3 during the simulation time interval which is from 70.5 seconds to 535 seconds (see Figure 1), whereas this system switches to another attractor whose maximum state value changes to be x_3 after a relatively shorter time interval (535, 908.3 seconds). From the experimental results, it is confirmed that the proposed bio-inspired model well inherits the mechanism of attractor selection and is able to dynamically capture the variation of the environmental conditions.

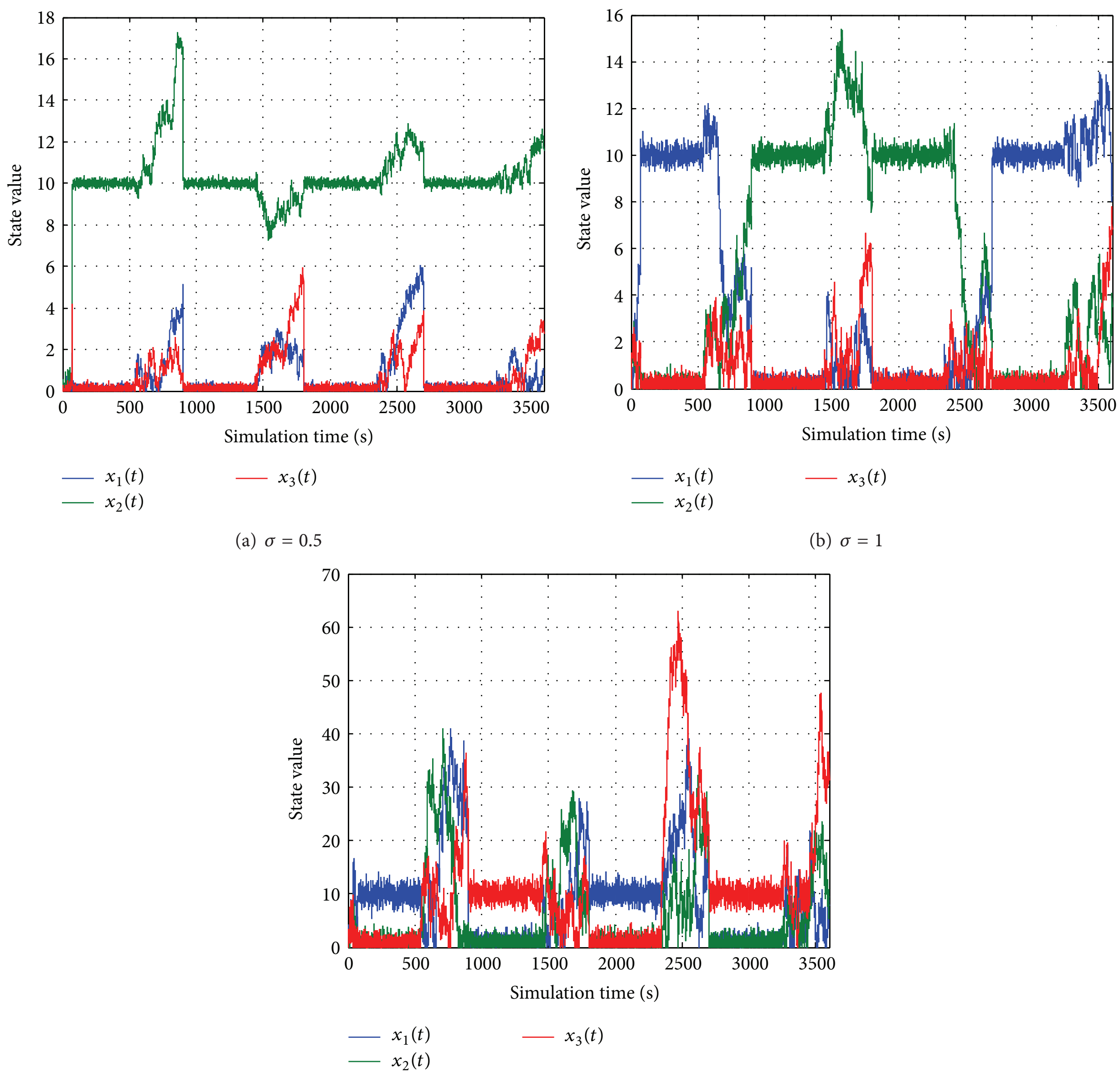
3. QoS-Oriented Handover Decision

This section gives the handover framework used in this work as well as the formulation for evaluating the terminal QoS satisfaction. Following this, we also present the detailed QoS-based handover decision scheme based on the extended attractor selection model.

3.1. Handover Decision Framework Based on the Extended Attractor Selection Model. Based on the extended attractor selection model, we propose a distributed handover decision making scheme framework. This framework is outlined in Figure 3. At each time period Δt, each individual mobile terminal $u_k \in$ User can perform the handover decision process independently. Because the extended attractor selection model only needs the information on the QoS requirements of the applications that belong to an individual terminal and the information on the current wireless network conditions, multiple terminals do not need to exchange their decision information with each other. In our scheme, the handover decision is made at the terminal side instead of the network side. Therefore, it is not necessary to deploy a centralized control entity for managing the handover process.

Under the distributed handover decision framework shown by Figure 3, each application of a mobile terminal firstly provides its QoS requirements to its terminal during each time period. In the meanwhile, the mobile terminal needs to sense the current network conditions by making some signaling messages interact with the networks through air interface. Then the utility of each application is evaluated by using the proposed utility function (the utility function is developed in Section 3.2), that is, quantifying the degree in which the QoS requirements of each application are satisfied by its current wireless link. Based on the QoS satisfaction degree of each application, we evaluate the terminal QoS satisfaction so as to map the degree of the terminal's QoS satisfaction to the activity α. Furthermore, the activity α is inputted to the extended attractor selection model and is used to drive the model to update the handover decision matrix given in (2) as well as determining the target networks by using (3). Finally, the handover can be done according to the determined target networks.

3.2. Quantification of Terminal's QoS Satisfaction. In this work, we take the upper and lower bounds of QoS requirements of each application into consideration. The QoS-related attributes considered include bandwidth, end-to-end delay, and packet loss ratio. The bandwidth, delay, and packet loss ratio are, respectively, indexed by the notations b, d, and p. Based on this, we denote the upper bound of bandwidth, delay, and packet loss ratio required by the application $s_i \in S_{u_k}$ as $U(s_i, x)$ ($x = b, d, p$) and the lower bound of those requirements as $L(s_i, x)$ ($x = b, d, p$). We assume that the application s_i is currently connected to the network $\text{net}_j \in$ NetSet. Additionally, the current conditions (including available bandwidth, transmission delay, and packet loss ratio) of the network net_j at time t are denoted as $C(\text{net}_j, x, t)$ ($x = b, d, p$). Then we evaluate the utility of each attribute



(a) $\sigma = 0.5$

(b) $\sigma = 1$

(c) $\sigma = 2.5$

FIGURE 2: Evaluation of the impact of the random noise variability on the dynamics system.

perceived by the application s_i with the linear normalization strategy as follows:

$$R(s_i, x, t) = \frac{\min\{U(s_i, x), C(\text{net}_j, x, t)\} - L(s_i, x)}{U(s_i, x) - L(s_i, x)}; \quad x = b,$$

$$R(s_i, x, t) = \frac{U(s_i, x) - \max\{L(s_i, x), C(\text{net}_j, x, t)\}}{U(s_i, x) - L(s_i, x)}; \quad x = d, p. \tag{12}$$

In addition, we adopt the weighted sum method to lump different $R(s_i, x, t)$ $(x = b, d, p)$ to one variable as follows:

$$G(s_i, t) = \sum_{x=b,d,p} \omega(s_i, x) \times f(R(s_i, x, t)), \tag{13}$$

where $\omega(s_i, x)$ is the positive weight corresponding to the QoS attribute x $(x = b, d, p)$, and they must satisfy the constraint $\sum_{x=b,d,p} \omega(s_i, x) = 1$. $f(\cdot)$ is a monotonously increasing function that can map $R(s_i, x, t)$ to $[0, 1]$. In this work, we formulate the function as the sigmoid form:

$$f(v) = \frac{1}{1 + \exp(-c_1 \times v + c_2)}, \tag{14}$$

where c_1 and c_2 are both positive real parameters.

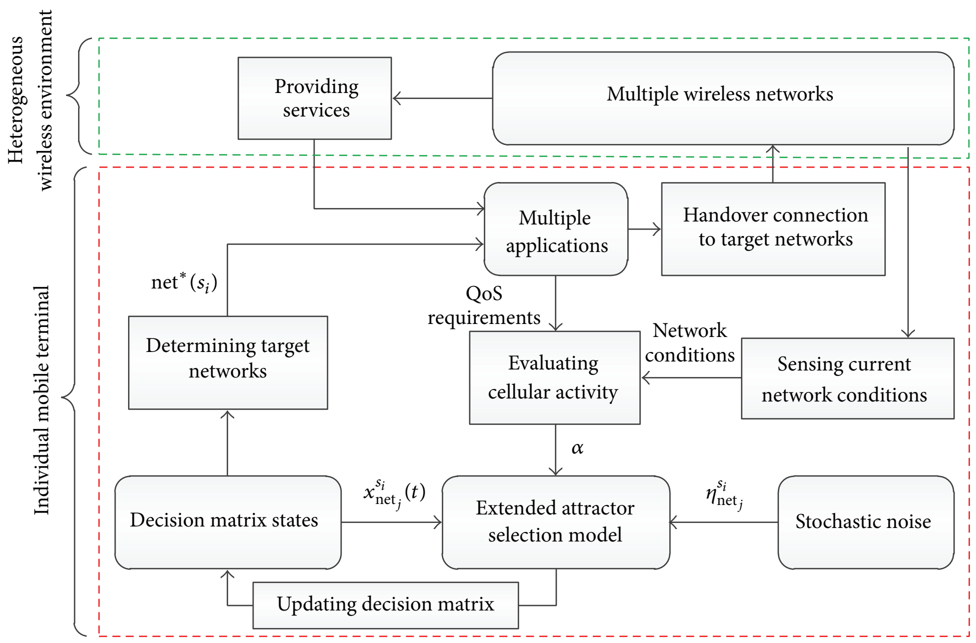

Heterogeneous wireless environment
Providing services
Multiple wireless networks
Individual mobile terminal
Multiple applications
Handover connection to target networks
net* (s_i)
QoS requirements
Network conditions
Determining target networks
Evaluating cellular activity
Sensing current network conditions
α
Decision matrix states
$x_{\text{net}_j}^{s_i}(t)$
Extended attractor selection model
$\eta_{\text{net}_j}^{s_i}$
Stochastic noise
Updating decision matrix

FIGURE 3: The handover decision framework based on the extended attractor selection model.

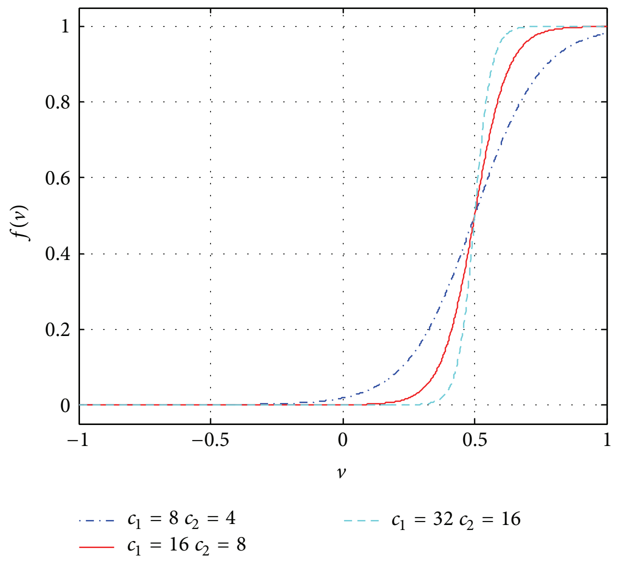


FIGURE 4: The variation of the adopted sigmoid function with different parameter settings.

In order to investigate the influence of c_1 and c_2 on $f(v)$ given in (14), we vary these parameters and plot the variation of the function with respect to different parameter settings. For simplicity but without loss of generality, we set $c_1 = 2c_2$ and obtain the results shown in Figure 4. The figure shows the slope of $f(v)$ with considering different values of c_1 and c_2. As we can see, the larger the parameters c_1 and c_2 are, the steeper the slope of $f(v)$ becomes. And the function value can stably stay at a relatively low (or high) level when the variable v approaches 0 (or 1). Because $R(s_i, x, t)$ represents the degree of the wireless network satisfying the application s_i in terms of the requirements on bandwidth, transmission delay and packet loss ratio, its value may change abruptly due to the dynamic nature of the wireless environment. On the basis of the results of Figure 4, we adopt $f(v)$ to map $R(s_i, x, t)$ to the interval $[0, 1]$ and fix the parameters c_1 and c_2 at 16 and 8, respectively, so that the weighted sum term $G(s_i, t)$ given in (13) is limited in $[0, 1]$ and its sensitivity is inhibited when $R(s_i, x, t)$ becomes either too much small or large.

On the other hand, the function value $G(s_i, t)$ comprehensively represents the degree of the QoS satisfaction of the application s_i. In order to evaluate the QoS satisfaction of the terminal u_k, we combine all of the $G(s_i, t)$ $(s_i \in S_{u_k})$ with the cumulative product strategy and yield

$$F(u_k, t) = \prod_{s_i \in S_{u_k}} G(s_i, t). \tag{15}$$

Since the terminal QoS satisfaction $F(u_k, t)$ may change suddenly at the instant time t along with the varying wireless environment, it is not feasible to directly set the activity α equal to $F(u_k, t)$ as the input of the extended attractor selection model. We further adopt the weighted moving averaging method to map $F(u_k, t)$ to α as follows:

$$\alpha = \frac{\int_{t-T}^{t} \lambda(\tau) \times F(u_k, \tau)\, d\tau}{T}, \tag{16}$$

where T is the fixed time window $(0 < T < t)$. $\lambda(\tau)$ $(\tau \in [t - T, t])$ is the time-dependent positive weight that is used to reflect the significance of $F(u_k, \tau)$ and needs to satisfy the constraint of $\int_{t-T}^{T} \lambda(\tau) d\tau / T = 1$. $\lambda(\tau)$ should be the increasing function of the time variable τ, because the closer to the current instant t the time variable τ is, the more significant the information reflected from $F(u_k, \tau)$ becomes. Therefore, for simplicity, we design $\lambda(\tau)$ as follows:

$$\lambda(\tau) = \frac{2\tau}{(2t - T)}. \quad (17)$$

It is easy to validate that (17) satisfies the constraint $\int_{t-T}^{T} \lambda(\tau) d\tau / T = 1$ and $\lambda(\tau) \geq 0$.

Based on (16), we can suppress the sensitivity of the activity α to the fierce changes in the dynamic wireless environment. Once α is obtained by (16), we treat α as the external input of the extended attractor selection model. Consequently, the model is driven to update the decision matrix in (2) so as to make the handover decision adaptively and automatically. Since each individual mobile terminal can independently process the extended attractor selection model and can perform the handover decision according to the aforementioned distributed framework, each individual terminal is essentially analogous to a cellular system.

4. Performance Evaluation

We perform some comparative simulations to evaluate the proposed bio-inspired handover decision scheme in this section through a discrete event simulator that we have developed in MATLAB with the object-oriented programming. Firstly, we present a typical heterogeneous wireless environment as the simulation scenario. And then we present the performance evaluation of our bio-inspired scheme and compare our scheme with the typical utility function based scheme with the simple additive weighting (SAW) [18].

4.1. Simulation Scenario. We consider a vehicular heterogeneous wireless environment where there exist three types of wireless networks including 3G cellular network (WCDMA), WiFi (IEEE 802.11n), and DSRC (IEEE 802.11p). This simulation scenario is given in Figure 5. The cellular network and DSRC are both assumed to be able to coverage the expressway denoted by the segment AE whose length is equal to 1000 meters. There are totally 3 WiFi access points (APs) deployed at the location of Points B, C, and D which are along the expressway AE. The coverage of WiFi APs is set to be 200 meters as illustrated in Figure 5. In the simulation, multiple vehicular terminals are stochastically generated and uniformly distributed on the expressway at the beginning of the simulation. The direction of these vehicles is from A to E and these vehicles are moving at a constant velocity during the simulation. In addition, each of these vehicular terminals runs three types of networking applications including the voice application, the video application, and the data stream. In addition, we set the time interval $\Delta t = 0.5$ seconds as the discrete period during which the decision matrix is updated at a time through processing the extended attractor selection model.

4.2. Simulation Settings. In this simulation, we use CDMA based cellular network and IEEE 802.11 based WLAN access networks including WiFi and DSRC. The network conditions related settings are referred to in the work [19] and are given in Table 1.

Table 1: Network conditions settings.

Network	Capacity (Mbyte/s)	Delay (ms)	Packet loss ratio (%)
Cellular	1.3	25	0.08
WiFi (AP1)	25	8	0.04
WiFi (AP2)	25	25	0.04
WiFi (AP3)	25	45	0.04
DSRC	27	50	0.03

It is worth pointing out that the network conditions are varying during simulation. Since the number of the connections to a network changes all the time and has a significant influence on the network resource, we assume that the amount of the connections to one network is the main factor to change this network's conditions over time. In order to simulate the dynamic nature of the heterogeneous wireless networks, we vary those network QoS attributes during simulation. Denote the amount of the applications that are currently connected to a network $\text{net}_j \in \text{NetSet}$ at time t as $\text{num}(\text{net}_j, t)$. For simplicity but without loss of generality, we simulate the time-dependent QoS attributes by using the following formulations:

$$\begin{aligned} C(\text{net}_j, b, t) &= \left\lfloor \frac{\text{Capacity}(\text{net}_j)}{\text{num}(\text{net}_j, t)} \right\rfloor, \\ C(\text{net}_j, d, t) &= \text{Delay}(\text{net}_j) \\ &\quad \times \left[1 + 0.5 \times f\left(\frac{\text{num}(\text{net}_j, t)}{\sum_{\text{net}_j \in \text{NetSet}} \text{num}(\text{net}_j, t)} \right) \right], \\ C(\text{net}_j, p, t) &= \text{Packet\_loss\_ratio}(\text{net}_j) \\ &\quad \times \left[1 + 0.5 \times f\left(\frac{\text{num}(\text{net}_j, t)}{\sum_{\text{net}_j \in \text{NetSet}} \text{num}(\text{net}_j, t)} \right) \right], \end{aligned} \quad (18)$$

where the notations $\text{Capacity}(\text{net}_j)$, $\text{Delay}(\text{net}_j)$, and $\text{Packet\_loss\_ratio}(\text{net}_j)$, respectively, present the capacity, delay, and packet loss ratio of the network net_j whose values are given in Table 1, $f(\cdot)$ is also the sigmoid function as shown by (14), and $\lfloor \cdot \rfloor$ is the floor function. From (18), it is obvious that the more the amount of the users connected to a network is, the worse the performance of this network will become. Thus, we are allowed to simulate the dynamic nature of the wireless network in the experiments.

Furthermore, Table 2 gives three typical applications and the upper and lower bounds of their QoS requirements. Their parameters settings are referred to in [20]. In our

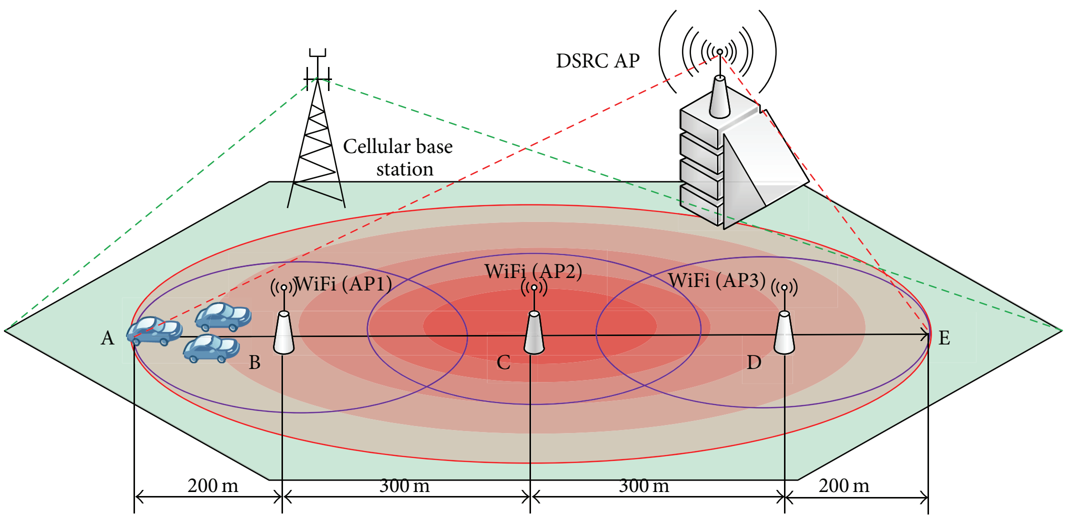


Figure 5: A simulation scenario.

Table 2: Applications settings.

QoS attributes	Requirements	Voice	Video	Data stream
Bandwidth (Kbyte/s)	$U(s_i, b)$	64	128	500
	$L(s_i, b)$	9	30	128
Delay (ms)	$U(s_i, d)$	150	150	120
	$L(s_i, d)$	0	0	0
Packet loss ratio (%)	$U(s_i, p)$	0.08	0.03	0.08
	$L(s_i, p)$	0	0	0

Table 3: Model settings.

Parameter	Value
μ	0
σ	1
β	5
m	5
n	3
c_1	16
c_2	8
T	25 s

Table 4: Settings on the weights.

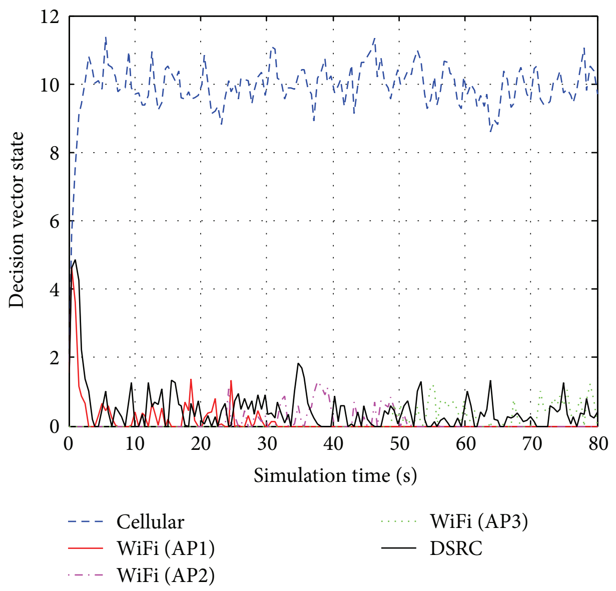
	$\omega(s_i, x)$	x		
		b (bandwidth)	d (delay)	p (packet loss ratio)
s_i	Voice	0.3	0.5	0.2
	Video	0.5	0.3	0.2
	Data stream	0.4	0.3	0.3

simulations, the total number of the applications run by each terminal is limited to three, and the applications of each terminal are generated from Table 2 at random. For example, a vehicular terminal may run all the three types of the applications including voice, video, and data stream while one other terminal may run two voice applications and a video application.

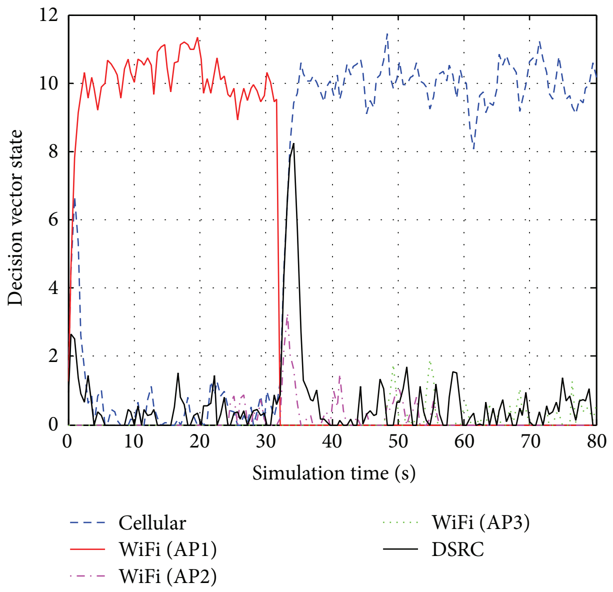
Additionally, we adopt the model settings illustrated in Table 3 for our extended attractor selection model. Since different applications are sensitive to different QoS attributes, the weights of the QoS attributes required by different applications are also different from each other. For instance, the voice application may require lower end-to-end delay and the video application needs more bandwidth for transmission. We use the detailed settings on $\omega(s_i, x)$ $(x = b, d, p)$ given in Table 4 for our experiments.

4.3. Numerical Evaluation. In order to analyze the process of the handover decision driven by the attractor selection mechanism from the extended attractor selection model, we initially set the number of the total vehicular terminals to be equal to 90 and the initial velocity of those terminals is set to be 45 km/h. We randomly choose one of those terminals and illustrate its relevant simulation results in Figure 6. This vehicular terminal has two types of applications, one of which is the voice-related application and the other two are the video-related applications. Subgraphs (a), (b), and (c) in Figure 6, respectively, show the variation of the decision vector state corresponding to each application against the simulation time. For example, in subgraph (a), the blue dashed line plots the variation of $x^{\text{voice}}_{\text{Celluar}}(t)$ that represents the variation of the fitness of the cellular network for the voice application against the simulation time.

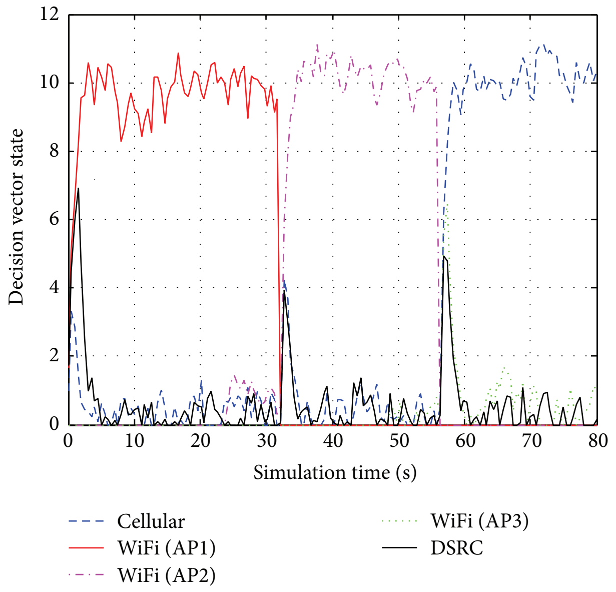
From these results, it is observed that the applications of this terminal are able to adapt its wireless access link according to the varying network conditions. According to subgraph (a), the voice application stably accesses to the cellular network during the whole simulation time, while the other two video applications switch their connections between different wireless networks in order to guarantee their QoS requirements. Even though these two applications are of the same type, their decision behaviors are different from each other. As shown in subgraphs (b) and (c), the accessing link of the



(a) The voice application



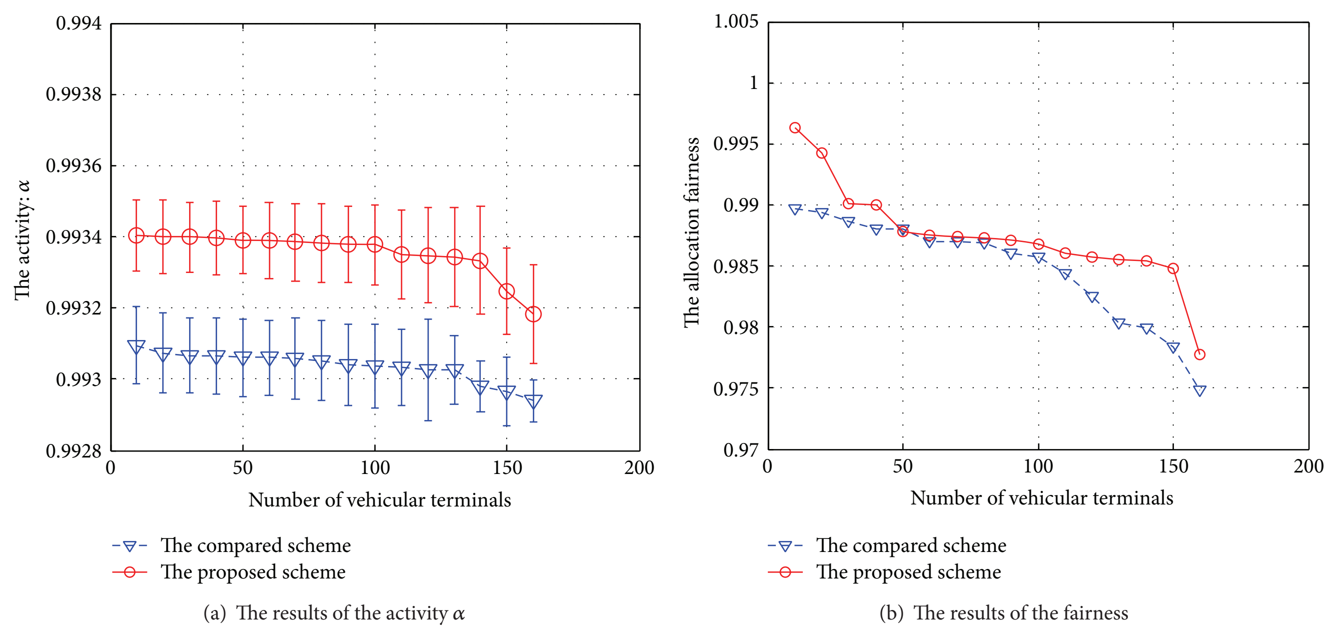
(b) One video application



(c) Another video application

Figure 6: The variation of the decision vector states of the different applications.

first video application switches to the cellular network from the first WiFi network when this terminal moves out of the coverage of the first access point (marked as “WiFi (AP1)” in Figure 5), while the second video application switches its connection to the second WiFi network and maintains this wireless connection for a while; after that, it also selects and stably connects to the cellular network. Since the heterogeneous network conditions are changing all the time, this terminal selects different appropriate networks for each of its applications according to the real-time network conditions so that it can ensure well QoS satisfaction. The average activity of this terminal reaches 0.98962. According to the definition of the activity illustrated by (16), the larger value of the activity indicates the higher degree of the terminal QoS satisfaction. Thus, this result implies that this terminal achieves a good QoS satisfaction during the process of handover decision making.



(a) The results of the activity α

(b) The results of the fairness

Figure 7: The simulation results under different vehicular terminals.

Next, we compare the simulation results obtained by our proposed bio-inspired handover decision scheme with those obtained by the utility function based scheme with SAW. Additionally, in order to perform the comparative evaluation in terms of the fairness of network resources allocation, we refer to the concept of the fairness index of the resource allocation among multiple entities in [21] and then define the following equation for calculating the fairness metric:

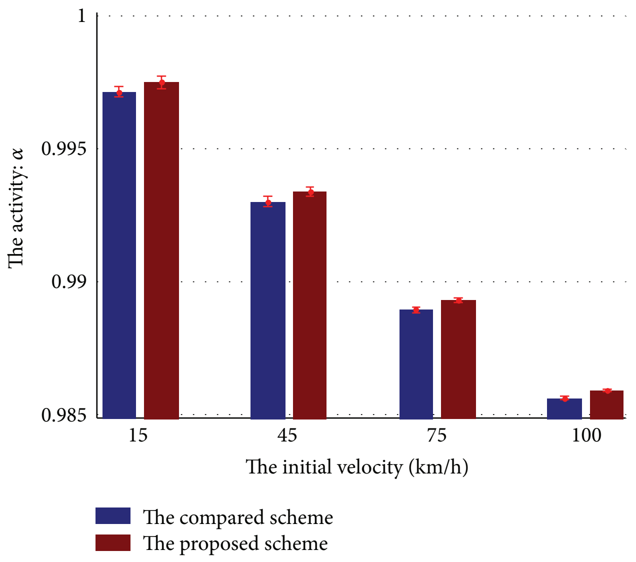
$$\text{The Allocation Fairness} = \frac{\left[\sum_{u_k \in \text{User}} \overline{\alpha}(u_k)\right]^2}{Q \times \left[\sum_{u_k \in \text{User}} (\overline{\alpha}(u_k))^2\right]}, \quad (19)$$

where Q is the number of the total vehicular terminals and $\overline{\alpha}(u_k)$ is the average activity corresponding to u_k that can be calculated by averaging all the values of the activity obtained at every time period Δt.

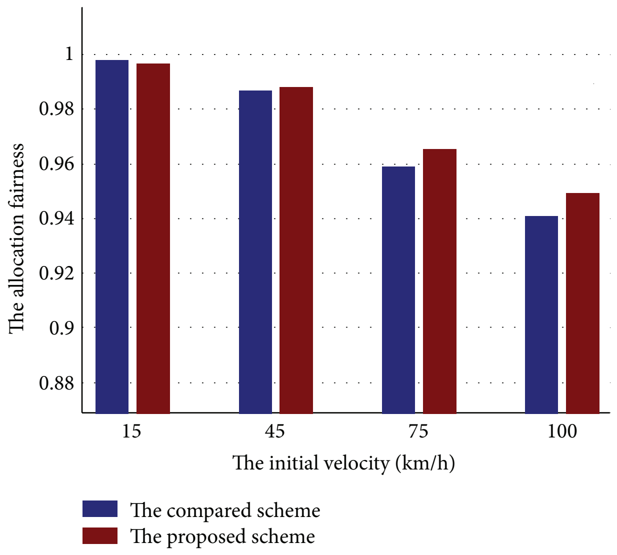
Firstly, we comparatively analyze the results of the two schemes under the specific simulation condition where the initial velocity is set to be 45 km/h and the number of total vehicular terminals discretely ranges from 10 to 160 so as to simulate the specific scenarios of different traffic densities. We calculate the average value and standard deviation of the activity α per individual terminal against different total numbers of vehicular terminals and the fairness of the overall network resources allocation according to (19). These results are illustrated in Figure 7. Since the amount of total vehicular terminals increases and the overall network resources are limited, the competition among multiple terminals becomes much fiercer so that the terminal QoS satisfaction and the fairness metric obtained by both schemes decreases along with increasing the amount of terminals. However, from subgraphs (a) and (b) in Figure 7, it can be found that the proposed scheme achieves a better activity and a better fairness of network resources allocation on average when compared with the results obtained by the compared scheme.

Furthermore, we perform the comparative simulations under different initial terminal velocity. We fix the number of vehicular terminals at 100 and vary the terminal velocity. The initial velocity is discretely set to be 15 km/h, 45 km/h, 75 km/h, and 100 km/h so as to simulate different mobility scenarios. We also calculate the average value and standard deviation of the activity α per individual terminal against different initial terminal velocity as well as the average value of the fairness metric. Figure 8 demonstrates those numerical results. Because the faster the vehicular terminal moves the shorter the duration when the applications of each terminal maintain their wireless links will last; a relatively high mobility may increase the times of switching wireless connection. This will reduce the efficiency of network resources allocation. Thus, the performance of both handover decision schemes degrades along with increasing the velocity. On the other hand, as shown in Figure 8, the activity on average obtained by our proposed scheme is larger than the compared scheme. Furthermore, the average degree of the fairness of the network resources allocation obtained by our scheme is larger than the compared scheme.

In fact, the utility function based handover decision with SAW selects the access link for each application in a deterministic manner. It ranks the networks based on the terminal utility and attempts to maximize every terminal's QoS satisfaction. Under the distributed framework, each application of these terminals tends to access the best network. Then, when large amount of applications accesses the same best network at the same time, this network will become congested and its performance will deteriorate rapidly. Consequently, this network becomes nonoptimal and its users will again switch to another same best network simultaneously. Thus, multiple



(a) The results of the activity α



(b) The results of the fairness

Figure 8: The simulation results under different initial velocity.

terminals may switch their connection between the optimal and the suboptimal networks much frequently. Therefore, the compared scheme potentially increases the handover times and reduces the efficiency of network resources allocation as well as the overall level of multiple terminals' QoS satisfaction. Unlike the deterministic optimization that behaves in a greedy manner, the attractor selection mechanism searches the attractor with stochastic optimization, that is, searching the optimal or suboptimal state with some certain noises. Although the attractor selection mechanism cannot guarantee that an application is connected to the best network all the time, it drives the multiple applications to select their appropriate access wireless links in a collaborative manner and makes these applications' connection adapt to the varying wireless network conditions, meanwhile meeting their QoS requirements to some extent. The mechanism drives multiple terminals to make handover decision in the way that is similar to the coexistence and self-adaptability of multiple cells behaving in a dynamic environment. Therefore, the bio-inspired attractor selection model is able to achieve a better performance from a global perspective.

5. Conclusions

In this paper, we propose a bio-inspired model for making handover decision in dynamic heterogeneous wireless environment. Our scheme provides a QoS-oriented handover solution for selecting an appropriate wireless network that can well satisfy the QoS requirements of each of the individual terminal applications in the dynamic context. For supporting adaptive and automatic decision making, we have introduced the attractor selection mechanism and proposed the distributed handover decision framework based on this bio-inspired model. Then, we model the activity parameter in terms of the individual terminal QoS satisfaction by a novel utility function, treat each individual terminal as a cellular system by analogy, and use the activity as the input of the extended attractor selection model to drive the process of updating the state value of the decision matrix as well as making handover decision.

The experimental results prove that the QoS-oriented handover decision scheme induced by the bio-inspired attractor selection model achieves better adaptation to the varying heterogeneous wireless environment and has better performance in terms of guaranteeing better QoS satisfaction and ensuring better fairness of network resources allocation when compared with the traditional utility function based scheme. In the future, we will extend our scheme by taking more decision factors into account such as the user profile related and terminal related decision factors, and we will validate the proposed scheme under some more complex heterogeneous environments where more dynamic characteristics are considered.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

This research is supported by the National Natural Science Foundation of China under Grant nos. 61103098 and 91118008, the National High Technology Research and Development Project (863 Project) under Grant no. 2012AA111901, the Beihang University Innovation & Practice Funding for Graduate under Grant no. YCSJ-02-2013-01, the Foundation of Key Laboratory of Road and Traffic Engineering of the Ministry of Education in Tongji University, Bagui scholarship project, the Natural Science Foundation of Guangxi under

Grant nos. 2011GXNSFA018154 and 2012GXNSFGA060003, the Science and Technology Foundation of Guangxi under Grant no. 10169-1, the Guangxi Scientific Research Project no. 201012MS274, the Funded Projects of Innovation Plan for Guangxi Graduate Education no. gxun-chx2013t18, the Natural Science Foundation of Guangxi under Grant no. 2013GXNSFAA019342, and the Guangxi University for Nationalities Project no. 2012QD017.

References

[1] E. Gustafsson and A. Jonsson, "Always best connected," *IEEE Wireless Communications*, vol. 10, no. 1, pp. 49–55, 2003.

[2] M. Zekri, B. Jouaber, and D. Zeghlache, "A review on mobility management and vertical handover solutions over heterogeneous wireless networks," *Computer Communications*, vol. 35, no. 17, pp. 2055–2068, 2012.

[3] X. H. Yan, Y. A. Sekercioglu, and S. Narayanan, "A survey of vertical handover decision algorithms in fourth generation heterogeneous wireless networks," *Computer Networks*, vol. 54, no. 11, pp. 1848–1863, 2010.

[4] F. Dressler and O. B. Akan, "A survey on bio-inspired networking," *Computer Networks*, vol. 54, no. 6, pp. 881–900, 2010.

[5] J.-B. Wang, M. Chen, X. Wan, and C. Wei, "Ant-colony-optimization-based scheduling algorithm for uplink CDMA nonreal-time data," *IEEE Transactions on Vehicular Technology*, vol. 58, no. 1, pp. 231–241, 2009.

[6] M. Meisel, V. Pappas, and L. Zhang, "A taxonomy of biologically inspired research in computer networking," *Computer Networks*, vol. 54, no. 6, pp. 901–916, 2010.

[7] C. Zheng and D. Sicker, "A survey on biologically inspired algorithms for computer networking," *IEEE Communications Surveys & Tutorials*, vol. 15, no. 3, pp. 1160–1191, 2013.

[8] H. Kitano, "Biological robustness," *Nature Reviews Genetics*, vol. 5, no. 11, pp. 826–837, 2004.

[9] H. Kitano, "Towards a theory of biological robustness," *Molecular Systems Biology*, vol. 3, no. 1, article 137, 2007.

[10] A. Kashiwagi, I. Urabe, K. Kaneko, and T. Yomo, "Adaptive response of a gene network to environmental changes by fitness-induced attractor selection," *PLoS ONE*, vol. 1, no. 1, article e49, 2006.

[11] I. Fukuyori, Y. Nakamura, Y. Matsumoto, and H. Ishiguro, "Control method for a robot based on the adaptive attractor selection model," in *Proceedings of the 4th International Conference on Autonomous Robots and Agents (ICARA '09)*, pp. 618–623, Wellington, New Zealand, February 2009.

[12] T. Iwai, N. Wakamiya, and M. Murata, "Error-tolerant coverage control based on bio-inspired attractor selection model for wireless sensor networks," in *Proceedings of the 10th IEEE International Conference on Computer and Information Technology (CIT '10)*, pp. 723–729, Bradford, UK, July 2010.

[13] Y. Koizumi, T. Miyamura, S. Arakawa, E. Oki, S. Kohei, and M. Murtata, "Adaptive virtual network topology control based on attractor selection," *Journal of Lightwave Technology*, vol. 28, no. 11, pp. 1720–1731, 2010.

[14] K. J. Leibnitz, N. Wakamiya, and M. Murata, "A bio-inspired robust routing protocol for mobile ad hoc networks," in *Proceedings of the 16th International Conference on Computer Communications and Networks (ICCCN '07)*, pp. 321–326, Honolulu, Hawaii, USA, August 2007.

[15] S. Kajioka, N. Wakamiya, and M. Murata, "Autonomous and adaptive resource allocation among multiple nodes and multiple applications in heterogeneous wireless networks," *Journal of Computer and System Sciences*, vol. 78, no. 6, pp. 1673–1685, 2012.

[16] J. Hasty, J. Pradines, M. Dolnik, and J. J. Collins, "Noise-based switches and amplifiers for gene expression," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 97, no. 5, pp. 2075–2080, 2000.

[17] M. B. Elowitz, A. J. Levine, E. D. Siggia, and P. S. Swain, "Stochastic gene expression in a single cell," *Science*, vol. 297, no. 5584, pp. 1183–1186, 2002.

[18] S. Sharna and M. Murshed, "Performance analysis of vertical handoff algorithms with QoS parameter differentiation," in *Proceedings of the 12th IEEE International Conference on High Performance Computing and Communications (HPCC '10)*, pp. 623–628, Melbourne, Australia, September 2010.

[19] J.-M. Kang, J. Strassner, S.-S. Seo, and J. W.-K. Hong, "Autonomic personalized handover decisions for mobile services in heterogeneous wireless networks," *Computer Networks*, vol. 55, no. 7, pp. 1520–1532, 2011.

[20] T. Kang, C. Hong, Y. Kim, and S. Kim, "A context-aware handoff management for seamless connectivity in ubiquitous computing environment," in *Proceedings of the International Conference on Pervasive Systems and Computing (PSC '06)*, pp. 128–134, Prague, Czech Republic, 2006.

[21] R. Jain, D. M. Chiu, and W. R. Hawe, "A quantitative measure of fairness and discrimination for resource allocation in shared computer system," Eastern Research Laboratory, Digital Equipment Corporation, 1984.

Hindawi Publishing Corporation
Journal of Applied Mathematics
Volume 2014, Article ID 197252, 9 pages
http://dx.doi.org/10.1155/2014/197252

Research Article

Functional Verification of High Performance Adders in Coq

Qian Wang,[1] Xiaoyu Song,[2] Ming Gu,[1] and Jiaguang Sun[1]

[1] *School of Software, Tsinghua University, Beijing 100084, China*
[2] *Department of ECE, Portland State University, Portland, OR 97207, USA*

Correspondence should be addressed to Qian Wang; qw04ster@gmail.com

Received 3 January 2014; Accepted 10 February 2014; Published 10 April 2014

Academic Editor: Krishnaiyan Thulasiraman

Addition arithmetic design plays a crucial role in high performance digital systems. The paper proposes a systematic method to formalize and verify adders in a formal proof assistant Coq. The proposed approach succeeds in formalizing the gate-level implementations and verifying the functional correctness of the most important adders of interest in industry, in a faithful, scalable, and modularized way. The methodology can be extended to other adder architectures as well.

1. Introduction

Demonstrating the functional correctness of an arithmetic implementation is a challenging topic which has lasted for several decades. Testing and simulation, as the traditional methods, have won good reputation and have been employed extensively in industry. When dealing with large scale designs, these methods may find counterexamples but could not assert if a design is correct because the exhaustivity is impractical.

As an alternative, formal methods have been increasingly adopted to validate the arithmetic implementations. A main branch of formal methods is model checking, which is recognised by its automation and succeeds in numerous industrial applications. However, the inherent state explosion problem prevents it from scaling to large scale designs.

Another branch of verification is theorem proving, which is no longer restricted by the scale as model checking, testing, and simulation. The main problem restricting theorem proving to be widespread is that it requires strong logic backgrounds and heavy user interactions. Nevertheless, there appear quite a few successful applications by different theorem provers. By Boyer-Moore, a microprocessor is verified in [1], and an N-bit comparator as well as mean-value circuits are verified in [2]. By HOL, a ripple carry adder and a sequential device are verified in [3], and an ATM switch fabric is verified in [4]. By Coq, a sequential multiplier is verified in [5], and an asynchronous transfer mode switch fabric is verified in [6].

The main effort of this work is to propose a holistic methodology to formalize and verify adders in Coq [7]. Adders are chosen because they are the most fundamental arithmetic units widely employed in various advanced digital systems, such as IBM POWER6, whose correctness depends significantly on the correctness of its addition subcomponents. This methodology provides a uniform way to formalize and verify various implementations of arithmetic addition, and it is applied in this work to formalize and verify primary and high speed adders of interest in industry, including Carry Look-ahead Adder (CLA), Ling Adder (LA), and Parallel Prefix Adder (PPA).

Benefiting from the techniques of Coq, the methodology shares the following decent features.

(i) Scalability: the formalization of an adder is parameterized by a natural number (named *length*) and the correctness proof applies to any length.

(ii) Modularization: various verified adders are encapsulated as instances of an abstract module, which provides a uniform way to be reused in advanced arithmetic units. The formalization and verification of an advanced arithmetic unit can be accumulated from verified units ignoring their detailed implementations.

(iii) Fidelity: the adders are formalized by (recursive) functions, which have clear correspondences to the gate-level implementations of circuits. The addends and sum of an adder are formalized as vectors, which is a faithful model of arrays and provides meanwhile additional type checking ability to avoid potential misusing of inputs.

The rest of paper is organized as follows. Related works are introduced in Section 2. According to our knowledge, we verify not only most adders appearing in the literature, but also some for the first time by theorem proving. Section 3 explains our methodology in details by the example of ripple carry adder. Preliminaries are also introduced according to our needs. Some definitions and most proofs will not be presented in this paper, but they are available on the author's webpage (http://superwalter.github.io/dev/veriadder.zip). Sections 4 and 5 are devoted to LA and PPA, respectively.

2. Related Work

Compared to their extensive applications, the verification of primary adders by theorem proving is not at the fingertips. In particular, the formalization and verification of the Ling adder cannot be found in any literature. Reference [8] proves the correctness of RCA by formalizing adders with dependent types in Coq. Reference [9] proves the correctness of RCA by the higher-order logic with a reusable library for formalizing circuits. Reference [2] verifies RCA written in VHDL as well as other circuits by the higher-order logic. Reference [10] develops semiformal correctness proof of CLA or PPA. Reference [11] shows a pencil-and-paper proof of the general prefix adders, as well as the proof of related RCA. Furthermore, [12] formalizes and verifies these adders in Coq. By rewriting and induction, [13] provides the verification of PPA using powerlists. An algebra formalization of PPA and its correctness proof are presented in [14]. Besides applying it to formalize and verify most primary adders, our methodology also provides good features, which only appear partially in other literatures, but are never integrated together in any preview work, according to our knowledge.

3. A Holistic Methodology

Various kinds of adders are designed to provide relatively good performances for different circumstances, while they implement the same addition functionality. A holistic methodology is proposed in this work in order to capture all the different adders and provide desired good features.

3.1. A Unified Proof Structure. Basically, the methodology answers four questions:

(i) how to formalize the related data types;

(ii) which method is used to formalize an adder;

(iii) what should be proved;

(iv) how to organize formalizations and verifications for different adders.

These questions are answered by a uniform specification, utilizing the module system of Coq.

```
 (1) Definition mbadder (n: nat):=
 (2) data (S n)-> data (S n)-> bit->
     hyb (S n).
 (3) Definition mbadder_c n (f: mbadder n):=
 (4) forall (X Y: data (S n)) c,
 (5) |[X]| + |[Y]| + |c| = |(f X Y c)|.
 (6) Module Type GenAdder.
 (7) Parameter adder: forall n, mbadder n.
 (8) Axiom adder_correct: forall (n:nat),
 (9) mbadder_correct (@adder n).
(10) End GenAdder.
```

Lines 1 and 2 answer the first two questions. n, in line 1, is a parameter (name *length*) indicating the inherent nature of an adder: how many bits it can process. The input carry-in and output carry-out are formalized by Booleans (`bit`). The input addends and the returned sum are formalized by vectors of Booleans (`data m`), which are dependent types depending on the length m. `hyb m` is another dependent type standing for a tuple of a bit and a m-bit vector, which is used in line 2 for combining the carry-out and the sum. Thus, an adder is formalized as a function, taking two addends and a carry-in as inputs and returning a tuple of carry-out and sum. This function is normally recursively defined as shown later.

Lines 3, 4, and 5 answer the third question. The correctness of an adder is ensured by proving that the natural number denotations of the inputs and outputs are equivalent. In line 5, $|b|$ is the natural number denotation of a bit b. $|[v]|$ and $|(t)|$ are natural number denotations of the vector v and the result tuple t. Big endian is chosen to implement these two functions.

Lines 6–10 answer the last question. A general adder is formalized as an abstract module. The specification is assigned and the correctness is required. A verified adder should be its instance, like a Ripple Carry Adder (RCA).

3.2. An Example Explaining the Methodology. Carry Look-ahead Adder (CLA) improves RCA by computing all the carries in advance in order to reduce the significant delay. This is represented, in the formalization, by extending the general module with abstract functions P, G, and *carries* which are supposed to compute all the propagated carries, generated carries, and carries, respectively, according to the inputs.

```
(1) Module Type LookAheadAdder <: GenAdder.
(2) Parameter P: forall n, data n -> data
    n -> data n.
(3) Parameter G: forall n, data n -> data
    n -> data n.
(4) Parameter carries n: data (S n) -> data
    (S n) -> bit -> hyb (S n).
(5) Parameter adder: forall n, mbadder n.
```

(6) `Parameter adder_correct: forall n,`
   `mbadder_correct (@adder n).`
(7) `End LookAheadAdder.`

<: symbol in line 1 stands for the fact that this module should be an instance of the general verified adder. RCA is formalized according to the following equations:

$$c_{i+1} = (x_i \wedge y_i) \vee ((x_i \oplus y_i) \wedge c_i) = g_i \vee (p_i \wedge c_i), \quad (1)$$

$$s_i = x_i \oplus y_i \oplus c_i = p_i \oplus c_i. \quad (2)$$

Carry to each bit c_{i+1} in CLA is computed by iteratively unfolding c_i in (1) until c_0 which is an overall input bit as shown by the following example:

$$\begin{aligned} c_3 &= g_2 \vee (p_2 \wedge c_2) \\ &= g_2 \vee (p_2 \wedge (g_1 \vee (p_1 \wedge c_1))) \\ &= g_2 \vee (p_2 \wedge g_1) \vee (p_2 \wedge p_1 \wedge c_1) \\ &= g_2 \vee (p_2 \wedge g_1) \vee (p_2 \wedge p_1 \wedge (g_0 \vee (p_0 \wedge c_0))) \\ &= g_2 \vee (p_2 \wedge g_1) \vee (p_2 \wedge p_1 \wedge g_0) \vee (p_2 \wedge p_1 \wedge p_0 \wedge c_0). \end{aligned} \quad (3)$$

This process as well as definitions of P and G are formalized as follows:

(1) `Definition P n (X Y: data n):= X ⊕ Y.`
(2) `Definition G n (X Y: data n):= X ∧ Y.`
(3) `Definition carries n (X Y: data (S n))`
   `(cin: bit): hyb (S n).`
(4) `induction n as [|n rec].`
(5) `+ exact (bandor (Y ▷) (X ▷) cin, [cin]).`
(6) `+ set (recs:= rec (X ◁) (Y ◁)).`
(7) `exact (bandor (Y ▷) (X ▷) (recs`$_1$`),`
   `(recs`$_1$`)⋈(recs`$_2$`)).`
(8) `Defined.`

⊕ and ∧ in lines 1 and 2 and ∨ used later are extensions of logical Boolean operations ⊕, ∧, and ∨, iterating these operations on the elements at the same position of the two vectors. + symbols in lines 5 and 6 stand for the start of the two branches of the recursion where $n = 0$ or $n = m + 1$. The ▷ operators in line 5 return the leftmost element of a vector. Correspondingly, the ◁ operator in line 6 returns the rightmost n elements of a $(n+1)$-bit vector. $[b]$ is a vector with a single bit b. p_1 and p_2 represent the first and second objects of a tuple, respectively. The ⋈ operator in line 9 joins a bit and a n-bit vector to form a $(n + 1)$-bit vector.

The adder is defined as follows and its correctness is proved by induction on the length and reusing the correctness result of the full adder:

(1) `Definition adder: forall n, mbadder n.`
(2) `intros n X Y cin.`
(3) `set (cc:= carries (P X Y) (G X Y) cin).`
(4) `exact (cc`$_1$`, (P X Y) ⊕ (cc`$_2$`)).`
(5) `Defined.`
(6) `Theorem adder_correct: forall n,`
   `mbadder_correct (@adder n).`
(7) `Proof. induction n as [|n rec].`
   `... Qed.`

3.3. Features Provided by the Methodology. There are several benefits to the use of this methodology for the verification of adders.

3.3.1. Scalability. The formalization and verification of an adder is scalable to any data-width, because the parameterized length can be specified to arbitrary natural number. A 4-bit RCA can be obtained by the following:

(1) `Definition CLA4:= CLA 3.`
(2) `Corollary CLA4_correct: forall`
   `(X Y: data 4) c,`
(3) `|[X]| + |[Y]| + |c| = |(RCA4 X Y c)|.`
(4) `Proof. intros; apply CLA_correct. Qed.`

Notice that a 4-bit CLA is `CLA3`, because we require that the addends of the adders have at least one bit. The correctness proof of a CLA with a specified length follows straightforwardly from the proof of CLA with arbitrary length.

3.3.2. Modularization. Some high speed adders divide the input addends into different groups. Each group is calculated by a Carry Selected Adder (CSA) independently, and different groups will be concatenated together in order. Since the computation of CSA depends on the very late steps of input carry-in, such designs would have less propagated time, thus high performance. We formalize an abstract architecture for this kind of design, which illustrates the modularization of our method and may also contribute to verify complex adders in the future.

CSA takes an abstract verified adder as parameter and is also an instance of the general verified adder.

(1) `Module CSA (M: GenAdder) <: GenAdder.`
(2) `Definition adder n: mbadder n.`
(3) `intros X Y c.`
(4) `set (a1:= M.adder X Y true).`
(5) `set (a0:= M.adder X Y false).`
(6) `set (sum:= (dmap (band c) a1`$_2$`) ∨`
   `(dmap (band (¬c)) a0`$_2$`)).`
(7) `set (c':= (a1`$_1$` ∧ c) ∨ (a0`$_1$` ∧ (¬c))).`
(8) `exact (c', sum).`
(9) `Defined.`

```
(10) Theorem adder_c: forall n, badder_
     correct (@adder n).
(11) Proof. ... rewrite M.adder_c. ... Qed.
(12) End CSA.
(13) Module CSA_CLA:= CSA CLA.
```

Lines 2 to 10 define CSA. Two adders compute the sum and the carry-out with respect to carry-in *true* and *false* in lines 4 and 5, respectively. The multiplexer chooses the real sum and carry-out according to the actual carry-in in lines 6 and 7, since when the input carry is required. *dmap* in line 6 applies a function to each element of a vector. The correctness of CSA holds because the addition unites are correct; thus, CSA is an instance of the general adder. The parameterized module can be instantiated by any verified adders. Line 13 defines a CSA whose addition unites are specified to CLA.

```
(1) Module Type GroupAdder (M: GenAdder)
    <: GenAdder.
(2) Parameter part: list nat.
(3) Fixpoint adder_rec (n lens len: nat):
    (mbadder lens).
(4) destruct n.
(5) + exact (@M.adder lens).
(6) + specialize adder_rec with (1:=n)
(7) (lens:= pred (cur_index_abr n len))
    (2:=len).
(8) ....
(9) exact (cast_comb (combination
(10) (@M.adder (lens - (cur_index_abr n
     len)))
(11) (adder_rec)) (aux _ _ Hc3 Hc2)).
(12) Defined.
(13) Definition adder n:= adder_rec (sect n)
     n n.
(14) Lemma adder_correct: forall n, mbadder_
     correct (adder n).
(15) Proof.   ...   Qed.
(16) End GroupAdder.
```

The formalization and verification of this adder are quite complex due to the problem with the dependent types as described in [15, 16]; therefore, the unimportant details are omitted. The *part* in line 2 is a partition of the addends. This partition should be valid, which means the elements preserve strict order and do not exceed the total data-width. Lines 3 to 12 define the adder recursively by combining an adder with another which is combination of the remaining groups of adders obtained by recursion. *combination* in line 9 execute the combining operation. *cast_comb* in line 9 converts an adder with length m to an adder with length n taking the proof of $m = n$ as an argument. The initial values of this recursive function are specified in line 13. The correctness can be proved by the induction on the length of the partition and using the correctness result of combining correct adders.

The parameterized module can be instantiated by any verified adder. If it is instantiated by CSA, it is a verification of many popular high speed adders.

3.3.3. Fidelity. There are normally two ways to formalize the addends and sum of an adder in Coq, either by dependent type *vector* as in [6, 8] and this work or nondependent type *list* as in [12]. Both [6, 8] have explanations why dependent type is more proper for the verification of adders. Generally speaking, nondependent list is more proper for formalizing linked list, whose length can be obtained by computation, while dependent vector is more proper for formalizing array, whose length is inherent natural. The functionality of adders is formalized by interactively defined (recursive) functions which have clear correspondences to gate-level description of circuits.

4. Ling Adder

The Ling Adder (LA) was proposed by [17]. Instead of computing in advance all the *carries* as CLA, LA computes all the pseudo carries, the propagation of which have less fan-ins and fan-outs. With the proper grouping of the input addends, LA needs lesser levels of gates and consequently has better performance.

Similar to the propagated and generated carries, LA has new complementing signal k_i and previous stage propagate T_i, which are defined in (4) and (5) respectively as follows:

$$k_i = a_i \wedge b_i, \tag{4}$$

$$T_i = a_i \vee b_i. \tag{5}$$

The pseudo carries are defined recursively. According to our knowledge, [17] and other materials about LA define the pseudo carries without considering the case $i = 0$ as this paper does in (6b).

Consider

$$H_i = k_i \vee (H_{i-1} \wedge T_{i-1}) \quad i > 0, \tag{6a}$$

$$H_i = k_i \vee c_{in} \quad i = 0. \tag{6b}$$

Without this case, the default values of H_{-1} and T_{-1} are both *false*, and it is equivalent to our definition assuming that c_{in} is always *false*. More intuitively, that algorithm does not consider the carry-in to the least significant bit,, which restricts it to some special applications, such as the addition of two registers. We generalize it to provide general functionality of an adder. Sum is defined similarly to consider the carry-in to the least significant bit as follows:

$$s_i = (H_i \oplus T_i) \vee (k_i \wedge H_{i-1} \wedge T_{i-1}) \quad i > 0, \tag{7a}$$

$$s_i = (H_i \oplus T_i) \vee (k_i \wedge c_{in}) \quad i = 0. \tag{7b}$$

The abstract module of Ling extends the general one by adding signatures of k, T, and H.

```
(1) Module Type LingAdder <: GenAdder.
(2) Parameter K: forall n, data n -> data
    n -> data n.
(3) Parameter T: forall n, data n -> data
    n -> data n.
(4) Parameter H: forall n, data (S n)-> data
    (S n)-> bit-> data (S n).
(5) Parameter adder: forall n, mbadder n.
(6) Parameter adder_correct: forall n,
    mbadder_correct (@adder n).
(7) End LingAdder.
```

To compute the ith pseudo carry of H, the ith bit of K and the $(i-1)$th bit of T are needed. Therefore, the two parameters of H stand for vectors K and a left shift of T. The formalization of H assuming the correctness of the parameters is as follows:

```
(1) Definition H n (X Y: data (S n)):
    data (S n).
(2) induction n as [|n rec].
(3) + exact ([(X ▷) ∨ (Y ▷)]).
(4) + set (recs:= rec (X ◁) (Y ◁)).
(5) exact ((X ▷) ∨ ((Y ▷) ∧ (recs ▷))
    ⋈ recs).
(6) Defined.
```

H is defined recursively. Line 3 deals with the case $i = 0$. Lines 4 and 5 deal with the recursive case. *recs* is the last n bits of H by recursion, and *recs* ▷ stands for H_{n-1}.

LA is defined according to (7a) and (7b) using the definition of H.

```
(1) Definition adder n
(2) (X Y: data (S n)) (cin: bit): hyb (S n).
(3) set (KXY:= K X Y).
(4) set (TXY:= T X Y).
(5) set (Tshft:= shiftin cin TXY).
(6) set (Hc:= H KXY (Tshft ◁)).
(7) set (Hcshft:= shiftin true pc).
(8) set (sum:= (TXY ⊕ Hc) ∨ (KXY ∧
    (Hcshft ◁) ∧ (Tshft ◁))).
(9) exact ((TXY ▷) ∧ (Hc ▷), sum).
(10) Defined.
```

Since the ith bit of sum depends on the $(i-1)$th bit of H and T, they are shifted in lines 5 and 7. The reason why *cin* is shifted into T is explained above; *true* is shifted into H to ensure $T_{-1} \wedge H_{-1} = cin$ where H_{-1} and T_{-1} are the bits to be shifted in, respectively, and $T_{-1} = cin$. The carry-out of LA is (TXY ▷) ∧ (Hc ▷) which is equivalent to c_{out} as shown in

$$c_i = H_{i-1} \wedge T_{i-1}, \quad i \geq 0. \tag{8}$$

The formalization of (8) is complicated, but the proof is trivial by induction and case analysis. The correctness of LA follows by proving a lemma stating that the outputs of CLA and LA are the same with regard to arbitrary same inputs. This lemma is proved by induction with the result of (8).

```
(1) Lemma LA_CLA_eq: forall n (X Y: data
    (S n)) c_in,
(2) LAdder.adder X Y c_in = CLAdder.adder
    X Y c_in.
(3) Proof. induction n as [|n rec].... Qed.
(4) Theorem adder_correct: forall n
    (X Y: data (S n)) c,
(5) |[X]| + |[Y]| + |c| =
    |(LAdder·adder X Y c)|.
(6) Proof. intros; rewrite LA_CLA_eq.
    apply CLAdder.adder_correct. Qed.
```

Reference [18] proposed an extension of Ling's adder by the following equations:

$$D_{j:k} = G_{j:k} + P_{j:k} = G_{j:k+1} + P_{j:k}, \tag{9}$$

$$B_{j:k} = g_j + g + j - 1 + \cdots + g_k, \tag{10}$$

$$G_{j:i} = D_{j:k} \wedge \left(B_{j:k} + G_{k-1:i}\right), \tag{11}$$

where $G_{i:j}$ and $P_{i:j}$ are group propagated and generated carries which are defined later in Section 5. Equation (11) is also proved in this work.

5. Parallel Prefix Adder

CLA improves RCA by computing all the carries in advance as shown in (4). However, large fan-in and fan-out will be caused if all the carries c_i are computed this way especially when i is large. Parallel Prefix Adder (PPA) avoids this by the idea of divide-and-conquer, which provides an efficient way to compute all the parallel carries. Basic definitions are as follows:

$$c_{i+1} = G_{i:j} \vee \left(P_{i:j} \wedge c_j\right) \quad (j \leq i), \tag{12}$$

$$s_i = c_i \oplus P_i, \tag{13}$$

$$P_{i:j} = \begin{cases} P_i & i = j \\ P_i \wedge P_{i-1:j} & i > j, \end{cases} \tag{14}$$

$$G_{i:j} = \begin{cases} G_i & i = j \\ G_i \vee \left(P_i \wedge G_{i-1:j}\right) & i > j. \end{cases} \tag{15}$$

Due to the similarity between (14) and (15), only the formalization of (15) is shown as follows. An auxiliary function, defined recursively on the difference of i and j, is reluctantly introduced to define it in Coq.

```
(1) Definition GpG_rec n (gp gg: data (S n))
    (d i: nat): bit.
(2) revert i; induction d as [|d rec];
    intros i.
(3) + exact (nth (n-i) gg).
(4) + exact ((nth (n-i) gg) ∨
    ((nth (n-i) gp) ∧ (rec (pred i)))).
(5) Defined.
(6) Definition GpG n (X Y: data (S n)) i j:=
(7) GpG_rec X Y (i-j) i.
```

In line 1, the parameters gp and gg stand for the propagated and generated carry vectors. Another parameter d is the difference of i and j. Function nth k v returns the kth element of v from the leftmost bit indexed 0. *pred n* computes the predecessor of n.

To compute all the carries parallel in advance, the carry c_{i+1} should not depend on any c_k, where $i \geq k > 0$, except c_0 which is the overall carry-in. Therefore, carries of PPA are computed according to a variation of (12) as follows:

$$c_{i+1} = G_{i:0} \vee \left(P_{i:0} \wedge c_0\right), \qquad (16)$$

and different PPAs employ different parallel prefix methods to compute the group carries $G_{i:0}$ and $P_{i:0}$, for all $n \geq i \geq 0$, for the sake of high performance. To capture various PPAs in a uniform framework, an abstract module, which abstractly describes this method as *groups*, is employed as follows:

```
(1) Module Type GroupCarries.
(2) Parameter groups: forall n, data2
    (S n) -> data2 (S n).
(3) Axiom groups_correct: forall
    n (X Y: data (S n)),
(4) groups (P X Y, G X Y) =
    correct_groups (P X Y, G X Y).
(5) End GroupCarries.
```

*data*2 n, in line 3, is the dependent type of a tuple of vectors whose lengths are both n. Therefore, the parameter of *groups* stands for vectors of propagated and generated carries as shown in line 6. *groups_correct* in line 4 is the assumption that the *groups* function is correct. The correctness is represented as an extensional equality of another correct function and itself. In line 7, *correct_groups* is the correct function to compute the groups carries according to (14) and (15). Its correctness holds by, first, computing all the carries *correct_carries* according to this function and then proving that *correct_carries* are equivalent to the carries of CLA.

```
(1) Definition correct_carries (n: nat)
    (c_in: bool)
(2) (X Y: data (S n)): hyb (S n).
(3) set (PXY:= P X Y).
(4) set (GXY:= G X Y).
(5) set (bvGp:= correct_groups (PXY, GXY)).
(6) set (all_c:= shift_map c_in bvGp).
(7) exact (all_c ▷, all_c ◁).
(8) Defined.
(9) Lemma carries_correct: forall n
    (X Y: data (S n)) c_in,
(10) correct_carries c_in X Y =
    CLAdder.carries (P X Y) (G X Y) c_in.
```

shift_map, in line 2, is a compositional operation first iterating Equation (16) on all the $G_{i:0}$ and $P_{i:0}$ which are stored in the vectors of the first and projection of *bvGp* and then shifting the overall carry-in c_0 to get all the carries. Consider that the computation of $G_{i:j}$ depends on the subgroups of the group propagated carries $P_{i:m}$, the *fundamental carry operator* "$\circ$" as in [19] is used to compute the group propagated and generated carries simultaneously in function *correct_groups* and should be used in all implementations of function *groups*. Consider

$$(P, G) \circ \left(P', G'\right) = \left(P \wedge P', G \vee \left(P \wedge G'\right)\right). \qquad (17)$$

Function *correct_groups* can be taken as an instance of *groups* function and is only one particular implementation of *groups*, which is verified. There are many other implementations of the *groups* function based on the following lemmas which are proved by induction on the difference between i and m, using (14) and (15):

$$\begin{aligned} P_{i:j} &= P_{i:m} \wedge P_{m-1:j} \quad (j < m \leq i), \\ G_{i:j} &= G_{i:m} \vee \left(P_{i:m} \wedge G_{m-1:j}\right) \quad (j < m \leq i). \end{aligned} \qquad (18)$$

Equation (18) can be rewritten using $\circ$ operator in one equation. For all $j < m \leq i$,

$$\left(P_{i:j}, G_{i:j}\right) = \left(P_{i:m}, P_{m-1:j}\right) \circ \left(G_{i:m}, G_{m-1:j}\right). \qquad (19)$$

Equation (19) shows clearly that any group of group carries can be computed by its concatenation (or even overlapped) subgroups. And the proper dividing and conquering of the bits of input addends can implement *groups* function with high performance. PPA is such a family of adders differing only in the computation of the *groups* function; thus, a general PPA can be formalized and parameterized by module *GroupCarries*.

```
(1) Module PPAdder (Import M: GroupCarries)
    <: GenAdder.
```

```
(2) Definition adder n (X Y: data (S n))
    (c_in: bit): (hyb (S n)).
(3) set (GT0:= groups ((P X Y), (G X Y)).
(4) set (all_carries:= shift_map c_in (GT0₁)
    (GT0₂)).
(5) set (sum:= PC ⊕ (all_carries ◁)).
(6) exact (all_carries ▷, sum).
(7) Defined.
(8) Theorem adder_correct: forall n
    (X Y: data (S n)) c_in,
(9) |[X]| + |[Y]| + |c_in| = |(adder X Y c_in)|.
(10) Proof.
(11) intros n X Y c_in; unfold adder.
(12) rewrite CLAdder.adder_correct.
(13) unfold CLAdder.adder.
(14) rewrite <- carries_correct.
(15) unfold correct_carries.
(16) rewrite groups_correct; trivial.
(17) Qed.
(18) End PPAdder.
```

Line 5, uses the abstract function *groups* from the parameterized module *GroupCarries* to compute all the group carries in advance. *shift_map* function in line 7 implements the operation in (16). Lines 6 and 8 compute all the carries and the sum.

The correctness of PPA is proved based on the assumption *groups_correct* in the abstract parameterized module *GroupCarries*. Line 15 reformats the left part of the equation to the result of what CLA computes. Line 17 uses the result that the carries of CLA are identical to the carries computed by (14), (15), and (16). Finally, the assumption *groups_correct* is used to prove that *groups* computes the same result as (14) and (15) do.

The rest of this section will show, by the example of Kogge-Stone addition algorithm, how this general PPA applies to some specific ones. The algorithm formalized following [20], in which the algorithm is proposed. Other implementations of PPA can be formalized and verified similarly.

```
(1) Module Kogge_Stone <: GroupCarries.
(2) Fixpoint KS_PG_rec (n m: nat)
    (bvPG: data2 (S n)): (data2 (S n)):=
(3) match m with
(4) | 0 => bvPG
(5) | S m'=> let recur:= (KS_PG_rec m'bvPG)
    in
(6) data2_op1 recur (shiftin_group
    (power2 m')recur)
(7) end.
(8) Definition groups n (PG: data2 (S n)):=
(9) KS_PG_rec (S (log2 (S n))) PG.
(10) Theorem groups_correct: forall n
     (X Y: data (S n)),
(11) groups (P X Y, G X Y) = correct_groups
     (P X Y, G X Y).
(12) End Kogge_Stone_Group_Carry.
```

Lines 2 to 10 describe the main function to define the Kogge-Stone implementation of the *groups* function. m is a simple counter to indicate how many *stages* are needed and which should the logarithm of the data-width be. When initializing, the input *bvPG* stands for two vectors of the propagated and generated carries, respectively, for example, $P_i = P_{i:i}$ and $G_i = G_{i:i}$, for all $n \geq i \geq 0$. At any stage m, this function computes the group carries of maximum length 2^m. A 8-bit kogge-stone adder is taken as an example to illustrate this procedure. At stage 2 ($m' = 2$), the group carries of maximum length 4 has been computed according to line 8:

$$\begin{aligned} recur := \big(&[P_{7:4}, P_{6:3}, P_{5:2}, P_{4:1}, P_{3:0}, P_{2:0}, P_{1:0}, P_0],\\ &[G_{7:4}, G_{6:3}, G_{5:2}, G_{4:1}, G_{3:0}, G_{2:0}, G_{1:0}, G_0]\big). \end{aligned} \tag{20}$$

At the next stage ($m = 3$), as in lines 8 and 9, firstly *shiftin_group* function shifts both vectors in the tuple simultaneously $2^{m'}$ times with *true* and *false*, respectively. The result is represented by $recur'$:

$$\begin{aligned} recur' := \big(&[P_{3:0}, P_{2:0}, P_{1:0}, P_0, \text{true}, \text{true}, \text{true}, \text{true}],\\ &[G_{3:0}, G_{2:0}, G_{1:0}, G_0, \text{false}, \text{false}, \text{false}, \text{false}]\big). \end{aligned} \tag{21}$$

Secondly, *date2_op1* executes the fundamental operator in (17) with two operands *recur* and $recur'$, and the result is

$$\begin{aligned} \big(&[P_{7:0}, P_{6:0}, P_{5:0}, P_{4:0}, P_{3:0}, P_{2:0}, P_{1:0}, P_0],\\ &[G_{7:0}, G_{6:0}, G_{5:0}, G_{4:0}, G_{3:0}, G_{2:0}, G_{1:0}, G_0]\big). \end{aligned} \tag{22}$$

In line 11, *groups* function specifies that the stages needed are $\log_2(n+1)$, where n is the data-width.

The correctness theorem cannot be proved by induction on the data-width as normal, because Kogge-Stone implementation of *groups* function recurses on the stages, not the data-width as shown in the definition of *KS_PG_rec*.

Noticing that, in the theorem, the result of each side of equation is a tuple of vectors, the equality holds if and only if the corresponding elements are identical pairwise.

```
(1) Lemma data_eq_nth_eq_data2: forall n
    (gx gy: (data2 (S n))),
(2) (forall k, k < S n ->
(3) (nth k (fst gx), nth k (snd gx)) =
```

(4) `(nth k (fst gy), nth k (snd gy))) <->`

(5) `gx = gy.`

However, the result of *KS_PG_rec* changes with the stage m, the first thing to prove is an invariant of m stating how this function approaches the result of *correct_group* function gradually with the increasing of the stages. Suppose, without loss of generality, that

$$\begin{aligned} \mathit{correct\_groups}(X,Y) &:= \left([P_{n:0},\dots,P_0],[G_{n:0},\dots,G_0]\right), \\ \mathit{KS\_PG\_rec}(m,Z) &:= \left([P^m_{n:0},\dots,P^m_0],[G^m_{n:0},\dots,G^m_0]\right), \end{aligned} \tag{23}$$

for all m, $Z = (X, Y)$ and $n > 0$; then, for all $0 \le k \le n$,

$$\left(P^m_{k:0}, G^m_{k:0}\right) = \begin{cases} \left(P_{k:0}, G_{k:0}\right) & k < 2^m \\ \left(P_{k:(k+1-2^m)}, G_{k:(k+1-2^m)}\right) & k \ge 2^m. \end{cases} \tag{24}$$

With this invariant, the existence of the fixed points can be proved secondly, and the least fixed point should be $\log_2(n+1)$. For all $m \ge \log_2(n+1)$and $k \le n < 2^m$, $(P^m_{k:0}, G^m_{k:0}) = (P^{m+1}_{k:0}, G^{m+1}_{k:0}) = (P_{k:0}, G_{k:0})$. Function *groups*, which iterates *KS_PG_rec* function $\log_2(n+1)$ stages, computes the same result as function *correct_groups*, which is the correctness theorem. The whole proof of this theorem has been carried out in COQ, although they are expressed in an intuitive way here for better understandings of the readers.

Kogge-Stone adder can be combined by the general module of PPA and this specific module of Kogge-Stone methods to compute all the group carries, which provides not only the computation method but also the correctness proof.

(1) `Module Kogge_Stone <: GenAdder:=`

(2) `PPAdder Kogge_Stone.`

6. Conclusion and Future Work

In this work, we proposed a holistic methodology to formalize and verify primary adders (RCA, CLA, LA, and PPA) in theorem prover Coq. They are formalized using dependent types, higher-order recursion and module systems in order to provide fidelity, scalability, and modularization.

In particular, PPA is a family of adders sharing the same structure, only differing in the methods of parallel prefix computing. We provide a novel way to describe the general PPA and show how to use this general module to verify a specific PPA, exemplified by Kogge-Stone adder.

Other advanced arithmetic designs can be verified reusing the formalizations and verifications of this work in a combinational way, as we describe by the example of carry select adders.

All the work has been carried out in Coq. The whole development contains around 2,000 lines of Coq scripts. This number of scripts is only about one third of [12], which is another work dedicated to verify additional designs in Coq. This work used lesser scripts but verified more addition designs than [12].

This work can be continued in two directions. Advanced arithmetic designs, such as IBM POWER6, can be cumulately verified based on these verified adders. Since formalization in a constructive way is to have clear correspondence to gate-level descriptions of circuits, HDL codes can be generated from the verified designs, which may provide an alternative way for designing the correct arithmetic implementations.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

This work has been supported by the National Science Foundation of China Grant 61272002, the Tsinghua National Laboratory for Information Science and Technology (TNList) Cross-discipline Foundation 2011-9, the Major Research plan of the National Natural Science Foundation of China Grant 91218302, and the National Basic Research Program of China (973 Program) Grant 2010CB328003.

References

[1] W. A. Hunt Jr. and B. C. Brock, "The verification of a bit-slice ALU," in *Hardware Specification, Verification and Synthesis: Mathematical Aspects*, M. Leeser and G. Brown, Eds., vol. 408 of *Lecture Notes in Computer Science*, pp. 282–306, Springer, New York, NY, USA, 1990.

[2] D. Borrione, L. Pierre, and A. Salem, "Formal verification of VHDL descriptions in the prevail environment," *IEEE Design & Test of Computers*, vol. 9, no. 2, pp. 42–56, 1992.

[3] A. Camilleri, M. Gordon, and T. Melham, *Hardware Verification Using Higher-Order Logic*, Computer Laboratory, University of Cambridge, Cambridge, UK, 1986.

[4] P. Curzon, "Experiences formally verifying a network component," in *Proceedings of the 9th Annual Conference on Computer Assurance (COMPASS '94)*, Safety, Reliability, Fault Tolerance, Concurrency and Real Time, Security, pp. 183–193, Gaithersburg, Md, USA, July 1994.

[5] C. Paulin-Mohring, "Circuits as streams in Coq: verification of a sequential multiplier," in *Types for Proofs and Programs*, S. Berardi and M. Coppo, Eds., vol. 1158 of *Lecture Notes in Computer Science*, pp. 216–230, Springer, Berlin, Germany, 1996.

[6] S. Coupet-Grimal and L. Jakubiec, "Certifying circuits in type theory," *Formal Aspects of Computing*, vol. 16, no. 4, pp. 352–373, 2004.

[7] The Coq Development Team, "The Coq proof assistant, reference manual," Tech. Rep. version 8.4, INRIA, Roquencourt, France, 2012.

[8] T. Braibant, "Coquet: a coq library for verifying hardware," in *Certified Programs and Proofs*, J.-P. Jouannaud and Z. Shao, Eds., vol. 7086 of *Lecture Notes in Computer Science*, pp. 330–345, Springer, Berlin, Germany, 2011.

[9] G. J. Milne, *Formal Specification and Verification of Digital Systems*, McGraw-Hill, New York, NY, USA, 1993.

[10] J. T. O'Donnell and G. Rünger, "Functional pearl derivation of a logarithmic time carry lookahead addition circuit," *Journal of Functional Programming*, vol. 14, no. 6, pp. 697–713, 2004.

[11] F. Liu, Q. Tan, and G. Chen, "Formal proof of prefix adders," *Mathematical and Computer Modelling*, vol. 52, no. 1-2, pp. 191–199, 2010.

[12] G. Chen, "Formalization of a parameterized parallel adder within the Coq theorem prover," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 29, no. 1, pp. 149–153, 2010.

[13] D. Kapur and M. Subramaniam, "Mechanical verification of adder circuits using rewrite rule laboratory," *Formal Methods in System Design*, vol. 13, no. 2, pp. 127–158, 1998.

[14] R. Hinze, "An algebra of scans," in *Mathematics of Program Construction*, D. Kozen, Ed., vol. 3125 of *Lecture Notes in Computer Science*, pp. 186–210, Springer, Berlin, Germany, 2004.

[15] B. Barras, J. P. Jouannaud, P. Y. Strub, and Q. Wang, "CoQMTU: a higher-order type theory with a predicative hierarchy of universes parametrized by a decidable first-order theory," in *Proceedings of the 26th Annual IEEE Symposium on Logic in Computer Science (LICS '11)*, pp. 143–151, Ontario, Canada, 2011.

[16] Q. Wang and B. Barras, "Semantics of intensional type theory extended with decidable equational theories," in *Computer Science Logic (CSL '13)*, S. R. D. Rocca, Ed., vol. 23 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pp. 653–667, Schloss Dagstuhl—Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany, 2013.

[17] H. Ling, "High-speed binary adder," *IBM Journal of Research and Development*, vol. 25, no. 3, pp. 156–166, 1981.

[18] R. Jackson and S. Talwar, "High speed binary addition," in *Proceedings of the Conference Record of the 38th Asilomar Conference on Signals, Systems and Computers*, vol. 2, pp. 1350–1353, Asilomar, Calif, USA, November 2004.

[19] I. Koren, *Computer Arithmetic Algorithms*, Universities Press, Hyderabad, India, 2002.

[20] P. M. Kogge and H. S. Stone, "A parallel algorithm for the efficient solution of a general class of recurrence equations," *IEEE Transactions on Computers*, vol. 2, no. 8, pp. 786–793, 1973.

Hindawi Publishing Corporation
Journal of Applied Mathematics
Volume 2014, Article ID 592638, 10 pages
http://dx.doi.org/10.1155/2014/592638

Research Article

Adaptive Fault-Tolerant Routing in 2D Mesh with Cracky Rectangular Model

Yi Yang,[1] Meirun Chen,[2] Hao Li,[3] and Lian Li[1]

[1] *School of Information Science and Engineering, Lanzhou University, Lanzhou 730000, China*
[2] *School of Applied Mathematics, Xiamen University of Technology, Xiamen 361024, China*
[3] *Laboratoire de Recherche en Informatique, Bat 490, Universite Paris-Sud 11, 91405 Orsay Cedex, France*

Correspondence should be addressed to Yi Yang; yy@lzu.edu.cn

Received 7 February 2014; Accepted 9 March 2014; Published 7 April 2014

Academic Editor: X. Song

This paper mainly focuses on routing in two-dimensional mesh networks. We propose a novel faulty block model, which is cracky rectangular block, for fault-tolerant adaptive routing. All the faulty nodes and faulty links are surrounded in this type of block, which is a convex structure, in order to avoid routing livelock. Additionally, the model constructs the interior spanning forest for each block in order to keep in touch with the nodes inside of each block. The procedure for block construction is dynamically and totally distributed. The construction algorithm is simple and ease of implementation. And this is a fully adaptive block which will dynamically adjust its scale in accordance with the situation of networks, either the fault emergence or the fault recovery, without shutdown of the system. Based on this model, we also develop a distributed fault-tolerant routing algorithm. Then we give the formal proof for this algorithm to guarantee that messages will always reach their destinations if and only if the destination nodes keep connecting with these mesh networks. So the new model and routing algorithm maximize the availability of the nodes in networks. This is a noticeable overall improvement of fault tolerability of the system.

1. Introduction

In the last decades, the goal of many researchers was to study communication operations in networks with fixed topologies, including modeling architectures and routing algorithm of parallel computers and cluster or middle area communication networks (such as metropolitan networks covering a town or a small region). The quality of such networks strongly depends on correct and efficient execution of communication operations.

Direct networks [1] become a popular architecture for communication networks, especially in massively parallel computer system. In direct networks, nodes (computers) are connected to only a few nodes, that is, its neighbours, according to the topology of the networks and communicate with each other by exchanging messages. Moreover, the mesh structure is one of the most important topology of direct networks. Especially, low dimensional mesh networks, due to its low node degree, are more popular than the high dimensional mesh networks. Currently most of architecture of parallel computers is based on two-dimensional mesh topology, for example, Seitz et al. 1988 [2], Intel Touchstone DELTA [3, 4], and Intel paragon.

Several models based on direct networks have been studied ([5–9]), especially the two-dimensional mesh ([10–16], etc.) for communication operations. The purposes of these papers mainly focus on how to route messages in the two-dimensional mesh. Routing is the process to send messages from source nodes to destination nodes, passing some intermediate nodes. A very important aspect of message routing is its ability to route from a source node to a destination node, avoiding all faulty nodes or links.

Basically, there are two types of message routing:

(1) *deterministic routing* that is routing in which the routes between given pairs of nodes are determined in advance of transmission,

(2) *adaptive routing* that allows us to take any path between its source and its final destination; that is,

the path is adaptively constructed in the process of routing.

The deterministic routing algorithms are simple and ease of implementation, this is the advantage for deterministic routing. However, adaptive routing can reduce network latency and increase network throughput and the most attractive point is that it can tolerant more faults than deterministic routing [17]. Thus the latter one emerged as an attractive field. In most papers on this field, they often considered how to make a path between source and destination node pairs, avoiding the faulty nodes, and most work used the disconnected rectangular block fault model [11]. The disconnected rectangular blocks are composed of the faulty nodes and their neighboring nonfaulty nodes with the principle of maintaining rectangular shape. As a result, adaptive routing can tolerate faulty nodes by bypassing these rectangles. However, in order to maintain its rectangular shape, the block has to group some nonfaulty nodes inside, called unsafe nodes in these papers. Of course, these unsafe nodes will never be used until their corresponding blocks recovery, and the messages will never be sent to these nodes, while they should be (as illustrated in Figure 1).

Chien and Kim [18] present a partially adaptive algorithm for mesh networks. The basic idea is to use the algorithm to circumfuse any convex faulty regions. If faulty regions are not naturally convex, good nodes and links are marked as faulty until the regions become convex. However, once the faults are located on a boundary, in order to tolerate faults, all nodes form that boundary will become faulty. Boppana and Chalasani [10] use f-chain and f-ring, which is an extension of disconnected rectangular block fault model, to route the messages around them, and f-chain addresses the boundary problem in the Chien and Kim's paper. But the f-chain and f-ring may connect with each other; this makes the routing algorithm more complex than [18]. In [11], Su and Shin assume a node to be the basic fault element. They construct the blocks based only on the faulty nodes; thus they can only tolerate faulty nodes except the faulty links. Overall, the construction of these faulty regions is static; that is, once these regions are constructed, all nodes including the good ones in these regions cannot join in routing any more. The faulty regions are not self-adaptive; that is, if some of faulty nodes in these faulty regions are fixed well, then the faulty regions will be held as they were, but actually they can release some good nodes and become smaller ones keeping convex shape.

Adaptive fault-tolerance routing technologies are also using in WSN (Wireless Sensor Networks), MEMS (Micro-Electro-Mechanical Systems) and SoC (System on Chip) to increase the usability and robustness, as well as the whole performance. Most network topology adopted in those domains is 2D mesh. As a result, in recent years, there have been a number of researches focusing on fault-tolerance routing on wsn and Noc [19–22].

In this paper, we concentrate on the adaptive routing with fault-tolerant in two-dimensional mesh. Not only we do consider the situation of faulty nodes but also the situation of faulty links incident with any node. However, different from mentioned papers, the novel cracky rectangular block strategy introduced to tolerate faults can route messages both bypassing the cracky rectangular block and along the cracks in the rectangular block (just for a trope, actually they are routed along the connected links inside the faulty blocks). So we can route messages to the nodes both outside and inside the faulty blocks. This is a noticeable overall improvement of fault tolerability of the system. At the same time, the cracky rectangular block is fully self-adaptive. It can tolerate dynamic faults. For example, when some of faulty nodes or faulty links in a block are fixed well, the original block may become a smaller block or split to some smaller ones keeping their shape rectangular. Tolerating dynamic faults can enhance the run-time life of a multicomputer, thus increasing reliability.

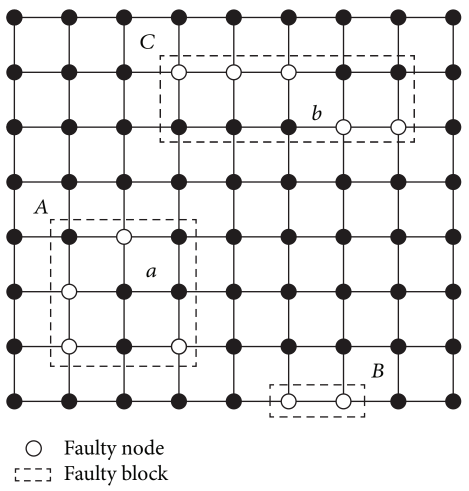


Figure 1: An example of disconnected rectangular blocks. Note that nonfaulty nodes, such as nodes a and b, in a block will never be used in routing any more.

The rest of this paper is organized as follows. Section 2 describes the basic routing algorithm in two-dimensional mesh. Section 3 introduces the cracky rectangular block strategy, including the cracky rectangular block model and the routing algorithm on it. This section also describes how the rectangular blocks adapt themselves depending on the situation of networks. Section 4 gives a proof that the message will be sent to any destination in the mesh as long as the mesh keep connecting. A conclusion will be given in Section 5, and it presents possible directions for future work.

2. Basic Routing Algorithm in Two-Dimensional Mesh

2.1. Two-Dimensional Mesh. It is convenient to represent a two-dimensional mesh with graph terminology. Let $G = (V, E)$ be undirected graph to represent a network. The set V of vertices of graph G represents nodes of the network. The set E of edges of graph G represents links between the nodes. Note that we keep using node and link in this paper.

The two-dimensional mesh $M(d_1, d_2)$, where $d_1 \geq 2$ and $d_2 \geq 2$ are positive integers, is defined as follows.

(i) A node X of $M(d_1, d_2)$ is represented by $X(x_1, x_2)$, $0 \leq x_1 \leq d_1 - 1$, and $0 \leq x_2 \leq d_2 - 1$.

(ii) There is a link between two different nodes $X(x_1, x_2)$ and $Y(y_1, y_2)$ if and only if $x_1 = y_1$ and $x_2 = y_2 \pm 1$ or $x_1 = y_1 \pm 1$ and $x_2 = y_2$. We denote this link by $\langle X, Y \rangle$.

For each k and k', with $0 \leq x \leq d_1 - 1$ and $0 \leq x' \leq d_2 - 1$, we call *row* k and *column* k' the subgraphs of $M(d_1, d_2)$, respectively, induced by nodes (x, k) and (k', x').

The *boundary* of $M(d_1, d_2)$ is the subgraph of $M(d_1, d_2)$ induced by the *rows* 0 and $d_1 - 1$ and the *columns* 0 and $d_2 - 1$.

Given any node v, let $\Gamma(v)$ be the set of nodes adjacent to v in $M(d_1, d_2)$ (called as *neighbours*). Given a nonboundary node $X = (x_1, x_2)$ of the two-dimensional mesh, the four neighbours of X are denoted by $W(X) = (x_1 - 1, x_2)$, $S(X) = (x_1, x_2 - 1)$, $E(X) = (x_1 + 1, x_2)$, and $N(X) = (x_1, x_2 + 1)$.

For each pair of nodes $X = (x_1, x_2)$ and $Y = (y_1, y_2)$, the distance between X and Y, denoted by $d(X, Y)$, is the length (number of links) of a shortest path between X and Y. We define the 1-distance and 2-distance between X and Y, respectively, by $d_1(X, Y) = |x_1 - y_1|$ and $d_2(X, Y) = |x_2 - y_2|$. From the above definition, we know that $d(X, Y) = d_1(X, Y) + d_2(X, Y)$.

2.2. A Basic Routing Function in Two-Dimensional Mesh. Consider a network $G = (V, E)$, in each node v, for each message m with final destination v_d, arriving on a link $\langle w, v \rangle$; we denote by $f_v(w, v_d) \subset \Gamma(v)$ the subset of v's neighbours bringing m closer to its destination if $v_d \neq v$; otherwise, the message is absorbed by v. Actually it is a routing function, this kind of routing is said to be local because it is independent of what happened in the rest of the network and can be computed locally by each router.

The basic routing function is a classical greedy routing function f_M in the two-dimensional mesh $M(d_1, d_2)$ as follows. Let $X = (x_1, x_2)$, $Y = (y_1, y_2)$ be two different nodes in $M(d_1, d_2)$. A message m with destination Y received by the router of X arriving from node X' is sent to a node of $f_X(X', Y)$, that is, a set of at most two nodes $\{V_1, V_2\}$ (and at least one node) defined as follows. There are at most two nodes $V_1 \in \{W(X), E(X)\}$ and $V_2 \in \{N(X), S(X)\}$ at distance $d(X, Y) - 1$ from Y. Moreover, when $|f_X(X', Y)| = 2$, if $d_1(X, Y) \geq d_2(X, Y)$ (resp., $d_1(X, Y) < d_2(X, Y)$), then the routing function will choose to send the message to V_1 (resp., V_2). If this is not possible (e.g., the link incident with the chosen node is faulty), then the routing function tries to send the message to V_2 (resp., V_1). Moreover, if both the links $\langle X, V_1 \rangle$ and $\langle X, V_2 \rangle$ are faulty, then the router of X can route m to any node of $\Gamma(X) \setminus \{V_1, V_2\}$.

2.3. Blocking Situation and Its Traditional Solution. Consider now that a unique message m is transmitted to the two-dimensional mesh $M(d_1, d_2)$. As we will show below, in case of some link faults which do not disconnect the network, using the basic two-dimensional routing function does not guarantee that m will reach its destination. It can be blocked in a part of the network.

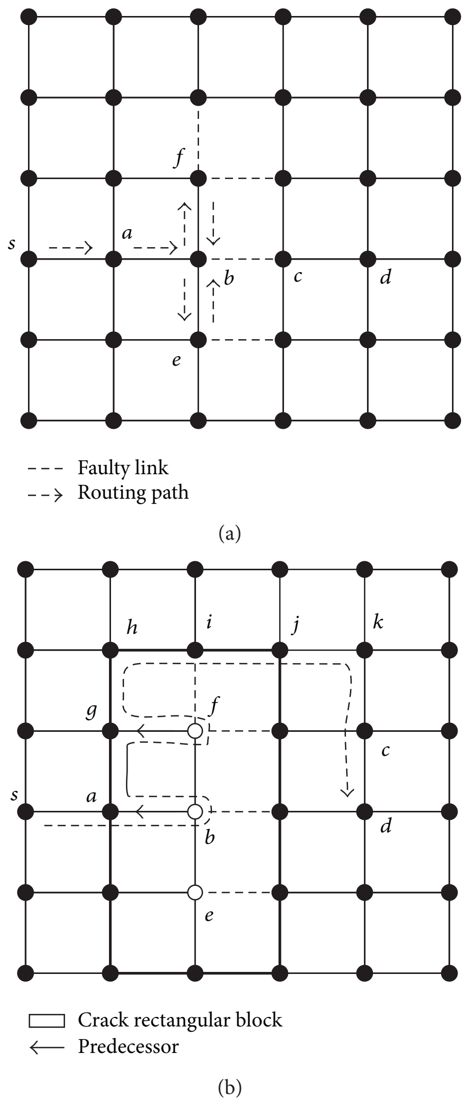


Figure 2: (a) Livelock situation in $M(d_1, d_2)$: node s tries to send a message m to node d, but m is in a livelock induced by nodes b, e, and f caused by the faulty links. (b) Cracky rectangular block solves the livelock problem, and the message m will be sent to its destination along the node sequences $s, a, b, a, g, f, g, h, i, j, k, c$, and d.

As shown in Figure 2(a), the basic routing function can unfortunately lead to blocking situations due to some properties of the structure of the faulty links. Clearly, in this example the message will always in the subgraph induced by nodes b, e, and f and will never reach its destination node d. Actually, this message m is in livelock situation, which keeps a message moving indefinitely without reaching the destination.

It is well known that the adaptive routing may cause livelock problems. Therefore routing without livelock is one of the most important design issues for communication operations in multicomputer systems (note that we only consider the livelock situation in this paper, and we can solve the deadlock problem with some sophisticated methods [1, 23, 24]). Contemporary, this livelock situation is well addressed by the traditional disconnected rectangular block faulty model (rectangular block for short). However, the usability and robustness of the mesh network will gradually decrease, while the number of faulty nodes increases in this model. As [25]'s experiment shows that the distribution of faulty nodes has the tendency to make the whole mesh to be one "big block." It can be seen from the experiments that, with the rectangular model, there is only one faulty block left when the faulty rate of nodes is 15 percent and the size of two-dimensional mesh is 100 × 100. In consequence the whole mesh becomes useless because this big faulty block occupies the entire mesh region, and we call this as "big block" problem. The novel cracky rectangular faulty block strategy, which we will introduce in the next section, makes full use of nonfaulty nodes/links in the mesh. All the nonfaulty nodes/links that would have been included in original rectangular faulty blocks now can become candidate routing nodes/links.

3. Adaptive Fault-Tolerant Strategy with Cracky Rectangular Block

In order to solve the livelock situation and the big block problem, we propose a novel strategy for fault-tolerant routing. We use the cracky rectangular block to avoid livelock and traverse block's every connecting internal node if needed. Therefore, we can transmit each message to any node not only outside of a block but also inside of the block like Figure 2(b), and the message can reach the inside nodes f, b, and e which are forbidden in the original rectangular block.

Formally, a rectangular block $C((l_1, h_1), (l_2, h_2))$ is a submesh $M(h_1 - l_1 + 1, h_2 - l_2 + 1)$ of the mesh $M(d_1, d_2)$ induced by the nodes $x = (x_1, x_2)$ with $l_i \le x_i \le h_i$, for each $i \in \{1, 2\}$. Let $u = (u_1, u_2)$ be a node. By definition, if, for each $i \in \{1, 2\}$, we have $l_i - 1 \le u_i \le h_i - 1$, then u belongs to the inside part of the rectangular block C. Else, if $i = 1, j = 2$ (or $i = 2, j = 1$) and $u_i \in \{h_i, l_i\}$ and $l_j \le u_j \le h_j$, then u belongs to the border of the rectangular block.

A *cracky rectangular block* (*cracky block* for short) is a rectangular block with spanning forest internal induced by all the connecting nodes inside of this block, all the roots of that forest belong to the border of the cracky block, and the spanning forest connects all the internal nodes to their roots if and only if those nodes still keep connecting.

Figure 3 presents two instances of the cracky block in a two-dimensional mesh, which are A and B, respectively. A is a general cracky block, while B is a cracky block which is induced by the faulty links on the boundary of the mesh, and it is an incomplete cracky block.

3.1. Construction of the Cracky Rectangular Block. Each node's activities are based on message-driven mechanism. There are two types of messages routed in mesh. One is *entity message* (message for short), which is routed between any node pair. The other one is *system message*, this type of message can only be sent between neighbours, and their contents are mainly about the status of themselves, such as the node's faulty degree and it's detailed situation of faulty links. The first one is the entity for computing or communication, and

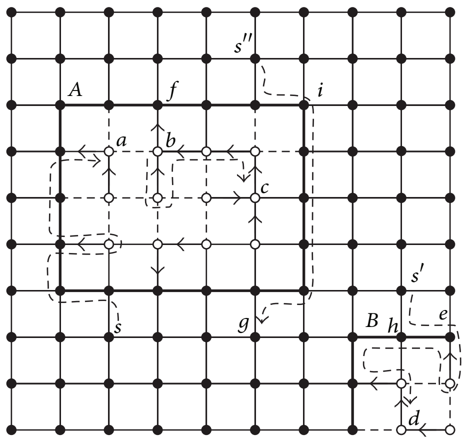


FIGURE 3: A and B are two cracky rectangular blocks. A is a general cracky block, while B is a cracky block which is induced by the faulty links on the boundary of the mesh, and it is an incomplete cracky block. As nodes a, b, and d in the cracky blocks A and B are faulty nodes, f and i are border nodes, and any node outside of A and B is good node. With the cracky rectangular block, not only good node s'' keeps connecting with another good node g as general, but also the good nodes s, s' can send messages to faulty nodes a, d, respectively; what is more, those faulty nodes can communicate with each other, such as nodes b and c.

the later one concentrates on maintaining the usability and robustness of networks; in other words, it is for constructing the cracky block when some faults occur in this mesh in order to avoid the livelock situation as mentioned above.

In the beginning, all the nodes work well; that is, there does not exist any faulty node or faulty link. Any node can both receive the message from any of its neighbours, and vice versa, of course, depending on the basic routing strategy. When some nodes or links are ruined because of some reasons, these failed nodes or nodes incident with failed links will judge their current status immediately, and then they send the system messages as soon as possible including their status to its connected neighbours to tell them what have happened in detail. For neighbours, once they receive the system messages, they judge their current status depending on their latest status and the received system messages at once. Of course, they will notice their connected neighbours about current status if and only if the current status is different from previous status. Finally, the construction of a stable cracky block is implemented by the above system messages exchange.

Before exposing our distributed algorithm to construct the cracky block, we will give some definitions first. For a two-dimensional mesh $M(d_1, d_2)$, the faulty degree of a node $X = (x_1, x_2)$ is the number of failed links incident with X, and we denote it by $f(X)$. From the observation of a cracky block, there are three types of nodes in a mesh network: *faulty node*, *good node*, and *border node*. A faulty node X belongs to the interior of a cracky block and $0 \le f(X) \le 4$, and oppositely, a good node allocates outside of any cracky blocks and $f(X) = 0$. Of course a border node belongs to the border of a cracky block with $0 \le f(X) \le 1$. For example, in Figure 3, a, b, and d in the cracky blocks A and B are faulty nodes, f

```
Input: $f(v)$: faulty degree of node $v$.
Output: $s(v)$: status of node $v$.
(1) procedure INITIAL_STATUS($v$)
(2)     if $f(v) = 0$ then
(3)         $s(v) \leftarrow 1$
(4)     else if $f(v) = 1$ then
(5)         $s(v) \leftarrow$ depending on Table 1
(6)     else if $f(v) \geq 2$ then
(7)         $s(v) \leftarrow \emptyset$
(8)     end if
(9)     notice_status($v$)
(10) end procedure
Input: $s(v)$: the current status of node $v$, $M(v)$: the system message received by node $v$.
Output: $s(v)$: the updated status of node $v$.
(11) procedure UPDATE_STATUS($v$)
(12)     if $s(v) \neq M(v) \cap s(v)$ then
(13)         $s(v) \leftarrow M(v) \cap s(v)$
(14)         notice_status($v$)
(15)     end if
(16) end procedure
Input: $s(v)$: the updated status of node $v$.
Output: $M(v)$: the system message to be sending to $N^*(v)$.
(17) procedure NOTICE_STATUS($v$)
(18)     if $s(v) = \emptyset$ then
(19)         send system message $M(v)$ to $N^*(v)$
(20)     else if $s(v) \in \{\{E\}, \{W\}\}$ then
(21)         send system message $M(v)$ to both $N(v)$ and $S(v)$ if exist
(22)     else if $s(v) \in \{\{N\}, \{S\}\}$ then
(23)         send system message $M(v)$ to both $E(v)$ and $W(v)$ if exist
(24)     end if
(25) end procedure
```

ALGORITHM 1: Let v be any node in the two-dimensional mesh, let $f(v)$ be the faulty degree, let $s(v)$ be the status, and let $M(v)$ be the content of received system message.

TABLE 1: Given a node v with $f(v) = 1$, judging its $s(v)$ according to its incident failed links.

$s(v)$	$f(v) = 1$
$\{N\}$	The link $\langle v, S(v)\rangle$ failed
$\{E\}$	The link $\langle v, W(v)\rangle$ failed
$\{S\}$	The link $\langle v, N(v)\rangle$ failed
$\{W\}$	The link $\langle v, E(v)\rangle$ failed

and i are border nodes, and any node outside of A and B is good node.

Given any node X, let $N^*(X)$ be the set of neighbors Y of X such that the link $\langle X, Y\rangle$ is not faulty. Moreover, we set an order in $N^*(X)$ as follows: $u_0, u_1, \ldots, u_{|N^*(X)|-1}$. Let X_p be the node who sends message to node X, and let X_s be the node who will receive the message sent by X.

We denote by $s(X)$ the status of X, and $s(X)$ is one of the elements of status set $STATUS = \{\{E\}, \{S\}, \{W\}, \{N\}, \{N, E\}, \{S, E\}, \{S, W\}, \{N, W\}\}$. The status of a node will indicate which type of node it is. In detail, there are two more status of X, which are empty set $\emptyset$ and universal set 1. And $s(X) = \emptyset$ shows that the node is a faulty node, $s(X) = 1$ identifies a good node, and if $s(X) \in STATUS$, then X must be a border node. For example, if $s(X) = \{N\}$, then the node X locates at the north border of a cracky block, like f in Figure 3, or if $s(X) = \{N, E\}$, then X is at the northeast corner, just like i. Totally, the system message can be sent to four neighbours, and X_s will be E, S, W, and N according to $E(X), S(X), W(X)$, and $N(X)$. A system message sent by node X is $M(X) = \{X_s\} \cup s(X)$ which includes the destination neighbour and sender's current status. For example, $M(X) = \{E, N\}$ means that this message will send to $E(X)$ and $s(X) = N$. We define an operation to implement the status judgment of a node who receives a novel system message. The corresponding algorithm to update the status for any node in mesh network is given by Algorithm 1.

At the beginning of the construction, every node should run the procedure initial_status respectively to make sure its status $s(v)$ according to its faulty degree $f(v)$. After finishing the above procedure, node will run the procedure notice_status to send system messages to neighbours according to its status $s(v)$. Once a neighbour node receives this type of message, it will run the procedure update_status to refresh its latest status. Actually, this process will be repeated until every node's status getting stable. Finally, there will emerge some cracky blocks in the mesh. For example, Figure 4 shows

```
Input: $v$: node $v$ is in a cracky rectangle block.
Output: $l(v)$: $\{h, f\}$, $pred(v)$: predecessor of $v$.
 (1) procedure HungNodeOnForest($v$)
 (2)     if $v$ is a good or border node then
 (3)         $l(v) \leftarrow h$
 (4)         return
 (5)     else if $v$ is a faulty node then
 (6)         $l(v) \leftarrow f$
 (7)     end if
 (8)     $u_i \in N^*(v)$
 (9)     check $l(u_i)$ for all $u_i$ until $\exists u \in N^*(v)$ s.t. $l(u) = h$
 (10)    $l(v) \leftarrow h$
 (11)    pred($v$) $\leftarrow u$
 (12) end procedure
```

Algorithm 2: Let v be a node in a cracky rectangle block, making it hung when it is possible.

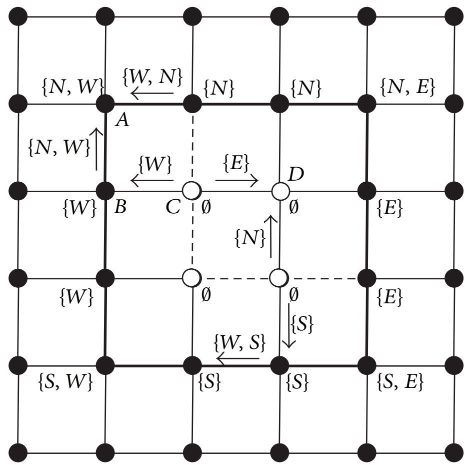


Figure 4: Construction of a cracky block depends on system message exchange. The dash line represents faulty link, and the bold line makes up the border of a cracky block. The arrow refers to the system message.

a distributed process to construct a cracky block. We just pick up four nodes, A, B, C, and D, to describe how the algorithm performs. During the first phase, node C will initial its status $s(C) = \emptyset$ because of $f(C) = 2$; meanwhile $s(A) = 1, s(B) = 1, s(D) = 1$ as a result of $f(A) = 0, f(B) = 0$, and $f(C) = 0$ separately. In the second phase, according to the algorithm only node C will send system message to its connected neighbours which are nodes B and D. Finally node A receives two system messages and will refresh its new status by $s(A) = (1 \cap \{N, W\}) \cap \{W, N\} = \{N, W\}$, so it will be the northwest of a cracky block for this moment. For nodes B and D, $s(B) = 1 \cap \{W\} = \{W\}$ and $s(D) = (1 \cap \{E\}) \cap \{N\} = \emptyset$ are a west border node and a faulty node. When the algorithm stops, there is a border of crack block as shown in Figure 4 by the bold line. The macroconstruction of a block depends on the microdistributed message exchange activities of relative nodes.

Then we will construct the spanning forest for the faulty nodes. We say that the faulty node is *hung* if and only if it chooses exactly one neighbor as predecessor. Denote by pred(v) the predecessor of v, and Succ(v) $= \{v' \in \Gamma(v) : v'$ is a faulty node and $v = \text{pred}(v')\}$. We consider an order over the elements of Succ(v). We denote by $\text{succ}_i(v)$ the ith element of Succ(v), with $1 \le i \le k_v = |\text{Succ}(v)|$. A node v is said to be *final* if Succ(v) $= \emptyset$. A node which is not hung is *free*. We denote by $l(v)$ these two boolean states $\{h, f\}$, which refers to the hung and free status for each node inside of the block. After running Algorithm 2, the spanning forest for the block will be accomplished; like A and B in Figure 3, every node inside of blocks A and B will find only one predecessor and be marked as hung.

3.2. The Cracky Rectangular Block Is Stable. A node v is said to be *stable* if pred(v)'s status can never change to a free status, during the running time of the algorithm. In particular the nodes of the cracky block are stable. A cracky block C is said to be stable if all the nodes belonging to C are stable.

To prove that the cracky block is stable, we assume that there exists a set S of nonstable nodes. If $S = \{v\}$, then v is free and can never become stable during the running time of Algorithm 2, so in this case there is no stable node in the neighborhood of v because otherwise v will choose this node as predecessor. Using the same argument for each node w of $N^*(v)$, there is no stable node in the neighborhood of w. But since the graph is connected, v is necessarily joined by a path to a node u of the border of the block. So we are in contradiction with the fact that the nodes of the border are stable. If $|S| \ge 2$, then since the graph is connected there exists at least one node u in S which is adjacent to a node $v \notin S$. Clearly, v is stable. From the second loop of the algorithm and since there is an order in the neighbors of u for the choice of its predecessor, there exists a step in the algorithm which leads the node u to choose v as predecessor. After this step, let $S \leftarrow S \setminus \{u\}$. Using the same arguments, after some steps of the algorithm, the set S would be empty. So all the nodes are stable.

3.3. Adaptive Routing with the Cracky Rectangular Blocks. In this section, we will give the global fault-tolerant routing

```
Input: $v$: node $v$ routing messages, $v_p$: the node who sends messages to node $v$.
Output: $v_s$: the node who will receive the messages.
(1) procedure ROUTING($v$)
(2)     if $v$ is a good node then
(3)         basic routing function with node $v$
(4)     else if $v$ is a faulty node then
(5)         if ($v$ is final) or ($v_p$) = $\text{succ}_{\kappa_v}(v)$ then
(6)             $v_s \leftarrow \text{pred}(v)$
(7)         else if $v_p = \text{succ}_i(v)$ and $i < \kappa_v$ then
(8)             $v_s \leftarrow \text{succ}_{i+1}(v)$
(9)         else if $v_p = \text{pred}(v)$ then
(10)            $v_s \leftarrow \text{succ}_1(v)$
(11)        end if
(12)    else if $v$ is a border node then
(13)        routing according to Table 2.
(14)    else if $v$ is a node belongs to the border of mesh then
(15)        if $v$ is final then
(16)            $v_s \leftarrow v_p$
(17)        else
(18)            $v_s \leftarrow \text{succ}_1(v)$
(19)            $\text{succ}_1(v) \leftarrow v_p$
(20)        end if
(21)    end if
(22) end procedure
```

ALGORITHM 3: The novel routing algorithm based on cracky rectangular blocks.

strategy. Primarily, once a message encounters a cracky block, this message will bypass the cracky block, which encloses the faulty nodes/links, along its border node in a clockwise (or counter-clockwise) manner. Especially, the message should traverse the interior spanning tree rooted with the border node by Depth-First-Search, while it bypasses the cracky block. Finally, the message will leave the cracky block from one of its corners which is the nearest from the destination and keeps going with the basic routing function; otherwise the message will be absorbed by the interior node which must be the destination node.

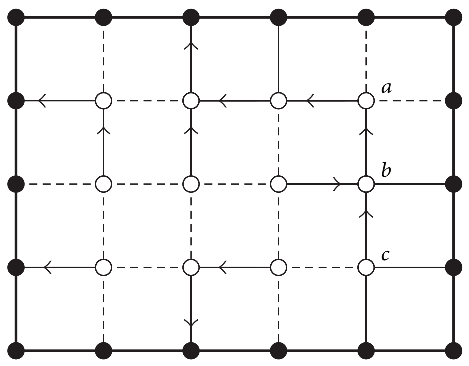
We now give the complete local routing function we run in each node v of $M(d_1, d_2)$, as shown in Algorithm 3. This algorithm is based on the basic routing function we have defined in Section 2.

The cracky rectangular block and the adaptive fault-tolerant algorithm make up the fault-tolerant strategy, and we can use Algorithm 3 to send a message from any connected node to arbitrary connected node. For example, in Figure 3, the good node s wants to send a message to the node a, but node a is a faulty node and locates interior the cracky block A, the algorithm will send this message along the path shown in the figure, and the faulty node b sending message to another faulty node c also can be accomplished by the algorithm; if a good node s'' wants to communicate with another good node g, the routing path will like the situation depicted in the figure.

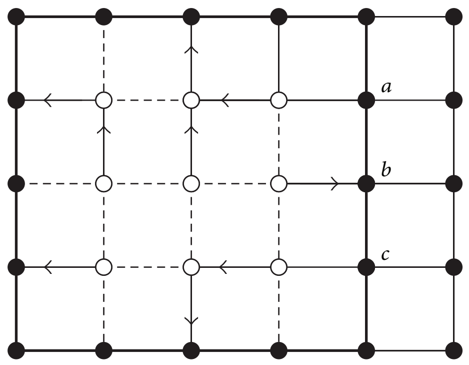
3.4. Self-Adaptive and Faulty Boundary Independency of Cracky Rectangular Block. For high performance and usability, the cracky blocks should be self-adaptive. As we know, the emergency of cracky blocks in a mesh is the result of nodes managing themselves distributedly and independently. The status of an isolated node is closely related to their neighbours. Therefore the size and shape of a block are dynamic according to faulty nodes. In other words, if some of the faulty nodes have been fixed, the original block may become a smaller one or split up into smaller ones. On the contrary, if some good nodes or links fail, there will be some new cracky blocks or some of the original cracky blocks grow huge as a result.

Given a two-dimensional mesh $M(d_1, d_2)$, let $v(v_1, v_2)$ be a faulty node in cracky block $C((l_1, h_1), (l_2, h_2))$. Let $X(x_i, v_1)$ with $l_1 \leq x_i \leq h_1$, and let $Y(v_2, x_j)$ with $l_2 \leq x_j \leq h_2$. There is a fact that when v has been repaired such that $f(v) = 0$, then we should make sure if $f(X) = 0$ (resp., $f(Y) = 0$). If they are, then X (resp., Y) may be cancelled from the block C and C will become four smaller ones at most. In addition, these new cracky blocks still keep stable. The cancelled row or column may becoming the new border belonging to those new cracky blocks, alternatively becoming the good ones outside any blocks, so they will still keeping hung, certainly their successors will also keeping hung. To implement the above, when a node with its incident links is fixed well, we just send a recovery signal to its four neighbours to rerun the procedure initial_status in Algorithm 1. Recursively, the recovery signal will be sent to nodes which connected with the faulty nodes received the signal until it meets the good node outside the cracky block.

For example, Figure 5(a) shows a cracky block, and a, c are two faulty nodes with $f(a) = 2$, $f(c) = 1$, and $f(b) = 0$. When the two nodes a and c have been repaired, they all changed to good nodes with $f(a) = f(c) = 0$. The cracky block will become like Figure 5(b), and a, b, and c become the



(a) Cracky block



(b) Cracky block after nodes a, b, and c fixed well

Figure 5: A sample of self-adaptive block.

new border and the cracky still keeps stable (note that we do not give the detail because of the page limitation).

Our adaptive fault-tolerant routing strategy is faulty boundary independency; that is, if there exists a fault occurring on the boundary of the mesh, the strategy is still running. Lines 14 to 21 in Algorithm 3 give a solution to this situation. When some of the boundary nodes of mesh have failed, then the corresponding cracky block will be constructed like B in Figure 3. As shown, it is an incomplete cracky block. If s' wants to send a message to d, the message will first go to node h according the basic routing function. Because h is the border node of B, the message will be sent to e which is a boundary node of the mesh. When the message traverses all the successors of e, it will be rebound to the node h to continue routing and finally find its destination. To sum up, the message will continue routing when it encounters a mesh boundary because of the rebound function.

4. The Cracky Rectangular Model Is Creditable

The next two propositions show that, with the above algorithm, each message will reach its destination, if a message arrives in a node of a cracky block C;

(i) if its destination is in C, it will reach it;

(ii) if its destination is out of C, it will leave C closer to its destination than before.

This is shown by the next two results.

Proposition 1. *Consider a cracky block C. By using Algorithm 3, if a message m has its destinations Y in C and if it arrives in a node of the cracky block, then it will reach its destination.*

Proof. Consider a subgraph S of the mesh induced by the nodes of a cracky block $C = C((l_1, h_1), (l_2, h_2))$. By definition of Algorithm 3, a message moving on S follows a circuit crossing each node at least once.

Consider a message m moving in the mesh, with destination $X(x_1, x_2) \in V(C)$, reaching a node $X' \in V(C)$. By definition of the routing function, since $l_1 \leq x_1 \leq h_1$ and $l_2 \leq x_2 \leq h_2$, then m will never want to take an arc out from the C. Thus, it follows the circuit of S induced by the algorithm. So, m will arrive in node X. □

Proposition 2. *Consider a cracky block C. By using the algorithm, if a message m has its destination Y outside C and if it arrives in node of the cracky block, then it will leave C and be closer to its destination than before.*

Proof. Let $m(Y)$ be a message with destination $Y(y_1, y_2)$. Suppose that $m(Y)$ moves from a node $X(x_1, x_2)$ to another node $X' = (x_1', x_2')$ by the dimension i according to the routing algorithm; that is, $|x_i - y_i| = \max\{|x_j - y_j| : 1 \leq j \leq 2\}$ and $|x_i - y_i| - 1 = |x_i' - y_i|$. We claim that, in the following routings, except the special case that m moves along a quasi-Hamiltonian cycle of a cracky block, the routing of m will never augment the i-dimension distance.

To prove the claim, it is clear that by the routing algorithm, if m do not meet a cracky block, it cannot move from a node $Z = (z_1, z_2)$ to another one $Z' = (z_1', z_2')$ with $|z_i - y_i| = |x_i' - y_i| = |x_i - y_i| - 1 = |z_i' - y_i| - 1$. Suppose now that m moves from $U' = (u_1', u_2')$ to meet a node $U = (u_1, u_2)$, with $|u_i - y_i| \leq |x_i - y_i| - 1$, of a cracky block $C = ((a_1, b_1), (a_2, b_2))$ by the dimension s and suppose that it leaves C by a node $W = (w_1, w_2)$ to a node $W' = (w_1', w_2')$ by the dimension t. According to the routing algorithm, without loss of generality, we may assume that the movement of m from X' to U' does not augment the s-dimension distance; that is, $|u_s' - y_s| \leq |x_s' - y_s|$. By the routing function, we have

$$\begin{aligned} \left|w_i' - y_i\right| &\leq \left|w_i - y_i\right| \leq \left|w_t - y_t\right| \leq \left|u_t - y_t\right| \leq \left|u_t' - y_t\right| \\ &\leq \left|u_s' - y_s\right| \leq \left|x_s' - y_s\right| \leq \left|x_s - y_s\right| \leq \left|x_i - y_i\right|. \end{aligned} \quad (*)$$

Assume that $|w_i' - y_i| = |x_i - y_i|$. We have all equalities in $(*)$, in particular $|u_t - y_t| = |u_s' - y_s|$. For any l, $1 \leq l \leq 2$, it follows that $|u_l - y_l| \leq |u_l' - y_l| \leq |u_s' - y_s| = |u_t - y_t|$, which implies that m would leave C from the node U by the t-dimension and hence $W = U$. This gives

$$\left|w_i' - y_i\right| \leq \left|w_i - y_i\right| = \left|u_i - y_i\right| \leq \left|x_i - y_i\right| - 1, \quad (1)$$

a contradiction. Therefore we have $|w_i' - y_i| \leq |x_i - y_i| - 1$ and the claim holds.

Consequently, if m leaves a cracky block C, it will never be sent back to C. □

Table 2: The routing table for border nodes of cracky blocks.

	v's border position is west	v's border position is east
$v_p = W(v)$	$(v_s \leftarrow E(v)$ and $\text{succ}_1(v) \leftarrow N(v))$ if (a) $v_s \leftarrow N(v)$ if (b)	$v_s \leftarrow \text{succ}_1(v)$
$v_p = S(v)$	$(v_s \leftarrow E(v)$ and $\text{succ}_1(v) \leftarrow N(v))$ if (a) $v_s \leftarrow N(v)$ if (b)	$(v_s \leftarrow W(v)$ and $\text{succ}_1(v) \leftarrow N(v))$ if (a) $v_s \leftarrow N(v)$ if (b)
$v_p = E(v)$	$v_s \leftarrow \text{succ}_1(v)$	$(v_s \leftarrow W(v)$ and $\text{succ}_1(v) \leftarrow S(v))$ if (a) $v_s \leftarrow S(v)$ if (b)
$v_p = N(v)$	$(v_s \leftarrow E(v)$ and $\text{succ}_1(v) \leftarrow S(v))$ if (a) $v_s \leftarrow S(v)$ if (b)	$(v_s \leftarrow W(v)$ and $\text{succ}_1(v) \leftarrow S(v))$ if (a) $v_s \leftarrow S(v)$ if (b)

	v's border position is south	v's border position is north
$v_p = W(v)$	$(v_s \leftarrow N(v)$ and $\text{succ}_1(v) \leftarrow E(v))$ if (a) $v_s \leftarrow E(v)$ if (b)	$(v_s \leftarrow S(v)$ and $\text{succ}_1(v) \leftarrow E(v))$ if (a) $v_s \leftarrow E(v)$ if (b)
$v_p = S(v)$	$(v_s \leftarrow N(v)$ and $\text{succ}_1(v) \leftarrow W(v))$ if (a) $v_s \leftarrow W(v)$ if (b)	$v_s \leftarrow \text{succ}_1(v)$
$v_p = E(v)$	$(v_s \leftarrow N(v)$ and $\text{succ}_1(v) \leftarrow W(v))$ if (a) $v_s \leftarrow W(v)$ if (b)	$(v_s \leftarrow S(v)$ and $\text{succ}_1(v) \leftarrow W(v))$ if (a) $v_s \leftarrow W(v)$ if (b)
$v_p = N(v)$	$v_s \leftarrow \text{succ}_1(v)$	$(v_s \leftarrow S(v)$ and $\text{succ}_1(v) \leftarrow E(v))$ if (a) $v_s \leftarrow E(v)$ if (b)

	v's border position is NE corner	v's border position is SE corner	v's border position is SW corner	v's border position is NW corner
$v_p = W(v)$	$v_s \leftarrow v_i$ if (c) $v_s \leftarrow S(v)$	$v_s \leftarrow v_i$ if (c) $v_s \leftarrow N(v)$	(d)	(d)
$v_p = S(v)$	$v_s \leftarrow v_i$ if (c) $v_s \leftarrow W(v)$	(d)	(d)	$v_s \leftarrow v_i$ if (c) $v_s \leftarrow E(v)$
$v_p = E(v)$	(d)	(d)	$v_s \leftarrow v_i$ if (c) $v_s \leftarrow N(v)$	$v_s \leftarrow v_i$ if (c) $v_s \leftarrow S(v)$
$v_p = N(v)$	(d)	$v_s \leftarrow v_i$ if (c) $v_s \leftarrow W(v)$	$v_s \leftarrow v_i$ if (c) $v_s \leftarrow E(v)$	(d)

(a): is not final.
(b): is final.
(c): let the destination node of the message be Y, if $\exists v_i \in \Gamma(v)$ and v_i's status is good, s.t. $d(v_i, Y) = d(v, Y) - 1$.
(d): using the basic routing function.

5. Concluding Remarks

In this paper, we propose a cracky rectangular fault block model for faulty-tolerant adaptive routing in two-dimensional mesh interconnection networks. This model improves the widely used rectangular model by taking into consideration the faulty links instead of faulty nodes in the process of constructing cracky blocks. It has been shown that we construct the spanning forest, which rooted with the border node, for all connected node in the cracky blocks. Thus the message can traverse all the nodes inside of the block by a kind of Depth-First-Search. As a result in the cracky block model, all faulty nodes that would have been useless now can be used for routing. Meanwhile the cracky block manages the size and scale in a self-adaptive mode; that is, the number or size of cracky block will gradually grow huge because of the increasing of faulty nodes/links, and contrarily, they will decrease for the fixed nodes/links. Based on the cracky block model, an algorithm is proposed to route message in the two-dimensional mesh without livelock. The novel strategy for fault-tolerant routing is faulty boundary independency, and it can apply the faulty occurring on the mesh boundary. The novel strategy for fault-tolerant routing improves the robustness and performance of two-dimensional mesh interconnection networks.

In the future, we will extend this strategy to multidimensional mesh networks, we have already testified that the construction method is suitable for multidimensional mesh networks, and then we will attempt to extend the routing algorithm to find a circuit on the multidimensional cracky blocks. In addition, we will add routing table to the cracky blocks to minimize the totally routing hops. These will come up in our next paper.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

The authors would like to thank the Natural Science Foundation of P. R. of China (61300230), the Key Science and Technology Foundation of Gansu Province (1102FKDA010), Natural Science Foundation of Gansu Province (1107RJZA188), and the Fundamental Research Funds for the Central Universities for supporting this research.

References

[1] M. N. Lionel and K. M. Philip, "A survey of wormhole routing techniques in direct networks," *Computer*, vol. 26, no. 2, pp. 62–76, 1993.

[2] C. L. Seitz, W. C. Athas, C. M. Flaig, A. J. Martin, J. Seizovic, and W. K. Su, "The architecture and programming of the ametek series 2010 multicomputer," in *Proceedings of the 3rd Conference on Hypercube Concurrent Computers and Applications*, pp. 33–37, 1988.

[3] Intel Corporation, *A Touchstone DELTA System Description*, 1991.

[4] S. L. Lillevik, "The touchstone 30 gigaflop DELTA prototype," in *Proceedings of the 6th Distributed Memory Computing Conference*, pp. 671–677, May 1991.

[5] P. T. Gaughan, B. V. Dao, S. Yalamanchili, and D. E. Schimmel, "Distributed, deadlock-free routing in faulty, pipelined, direct interconnection networks," *IEEE Transactions on Computers*, vol. 45, no. 6, pp. 651–665, 1996.

[6] Y. J. Suh, B. V. Dao, J. Duato, and S. Yalamanchili, "Software based fault-tolerant oblivious routing in pipelined networks," in *Proceedings of the International Conference on Parallel Processing*, pp. I101–I105, 1995.

[7] G. M. Chiu and S. P. Wu, "A fault-tolerant routing strategy in hypercube multicomputers," *IEEE Transactions on Computers*, vol. 45, no. 2, pp. 143–155, 1996.

[8] T. C. Lee and J. P. Hayes, "A fault-tolerant communication scheme for hypercube computers," *IEEE Transactions on Computers*, vol. 41, no. 10, pp. 1242–1256, 1992.

[9] A. C. Liang, S. Bhattacharya, and W. T. Tsai, "Fault-tolerant multicasting on hypercubes," *Journal of Parallel and Distributed Computing*, vol. 23, no. 3, pp. 418–428, 1994.

[10] R. V. Boppana and S. Chalasani, "Fault-tolerant wormhole routing algorithms for mesh networks," *IEEE Transactions on Computers*, vol. 44, no. 7, pp. 848–864, 1995.

[11] C. C. Su and K. G. Shin, "Adaptive fault-tolerant deadlock-free routing in meshes and hypercubes," *IEEE Transactions on Computers*, vol. 45, no. 6, pp. 666–683, 1996.

[12] C. Glass and L. Ni, "Maximally fully adaptive routing in 2D meshes," in *Proceedings of the International Conference on Parallel Processing*, vol. 1, pp. 101–104, August 1992.

[13] Y. M. Boura and C. R. Das, "Fault-tolerant routing in mesh networks," in *Proceedings of the International Conference on Parallel Processing*, pp. I106–I109, 1995.

[14] R. Libeskind-Hadas and E. Brandt, "Origin-based fault-tolerant routing in the mesh," in *Proceedings of the 1st International Symposium on High Performance Computer Architecture*, pp. 102–111, 1995.

[15] J. Wu, "Fault-tolerant adaptive and minimal routing in mesh-connected multicomputers using extended safety levels," *IEEE Transactions on Parallel and Distributed Systems*, vol. 11, no. 2, pp. 149–159, 2000.

[16] D. Xiang, J.-G. Sun, J. Wu, and K. Thulasiraman, "Fault-tolerant routing in meshes/tori using planarly constructed fault blocks," in *Proceedings of the International Conference on Parallel Processing*, pp. 577–584, June 2005.

[17] W. Chou, A. W. Bragg, and A. A. Nilsson, "The need for adaptive routing in the chaotic and unbalanced traffic environment," *IEEE transactions on communications systems*, vol. 29, no. 4, pp. 481–490, 1981.

[18] A. A. Chien and J. H. Kim, "Planar-adaptive routing: low-cost adaptive networks for multiprocessors," in *Proceedings of the19th International Symposium on Computer Architecture*, pp. 268–277, May 1992.

[19] M. Ebrahimi, M. Daneshtalab, J. Plosila, and F. Mehdipour, "MD: minimal path-based fault-tolerant routing in on-chip networks," in *Proceedings of the 18th Asia and South Pacific Design Automation Conference (ASP-DAC '13)*, pp. 35–40, Yokohama, Japan, January 2013.

[20] M. Ebrahimi, M. Daneshtalab, and J. Plosila, "High Performance Fault-Tolerant Routing Algorithm for NoC-Based Many-Core Systems," in *Proceedings of the 21st Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP '13)*, pp. 462–469, Belfast, UK, February-March 2013.

[21] M. Ebrahimi, H. Tenhunen, and M. Dehyadegari, "Fuzzy-based adaptive routing algorithm for networkson-chip," *Journal of Systems Architecture*, vol. 59, no. 7, pp. 516–527, 2013.

[22] S. S. Alamian, R. Fallahzadeh, S. Hessabi, and J. Alirezaie, "A novel test strategy and fault-tolerant routing algorithm for NoC routers," in *Proceedings of the 17th CSI International Symposium on Computer Architecture and Digital Systems (CADS '13)*, pp. 133–136, Tehran, Iran, October 2013.

[23] X. Lin and L. M. Ni, "Deadlock-free multicast wormhole routing in multicomputer networks," in *Proceedings of the 18th International Symposium on Computer Architecture*, pp. 116–125, May 1991.

[24] W. J. Dally and C. L. Seitz, "Deadlock-free message routing in multiprocessor interconnection networks," *IEEE Transactions on Computers*, vol. 36, no. 5, pp. 547–553, 1987.

[25] D. Wang, "A rectilinear-monotone polygonal fault block model for fault-tolerant minimal routing in mesh," *IEEE Transactions on Computers*, vol. 52, no. 3, pp. 310–320, 2003.

Hindawi Publishing Corporation
Journal of Applied Mathematics
Volume 2014, Article ID 294870, 10 pages
http://dx.doi.org/10.1155/2014/294870

Research Article

Total Variation Based Perceptual Image Quality Assessment Modeling

Yadong Wu,[1,2] Hongying Zhang,[3] and Ran Duan[3]

[1] *School of Computer Science and Technology, Southwest University of Science and Technology, Mianyang 621010, China*
[2] *Fundamental Science on Nuclear Wastes and Environmental Safety Laboratory, Southwest University of Science and Technology, Mianyang 621010, China*
[3] *Robot Technology Used for Special Environment Key Laboratory of Sichuan Province, School of Information and Engineering, Southwest University of Science and Technology, Mianyang 621010, China*

Correspondence should be addressed to Yadong Wu; wyd028@163.com

Received 12 February 2014; Accepted 10 March 2014; Published 1 April 2014

Academic Editor: X. Song

Visual quality measure is one of the fundamental and important issues to numerous applications of image and video processing. In this paper, based on the assumption that human visual system is sensitive to image structures (edges) and image local luminance (light stimulation), we propose a new perceptual image quality assessment (PIQA) measure based on total variation (TV) model (TVPIQA) in spatial domain. The proposed measure compares TVs between a distorted image and its reference image to represent the loss of image structural information. Because of the good performance of TV model in describing edges, the proposed TVPIQA measure can illustrate image structure information very well. In addition, the energy of enclosed regions in a difference image between the reference image and its distorted image is used to measure the missing luminance information which is sensitive to human visual system. Finally, we validate the performance of TVPIQA measure with Cornell-A57, IVC, TID2008, and CSIQ databases and show that TVPIQA measure outperforms recent state-of-the-art image quality assessment measures.

1. Introduction

Visual quality evaluation has numerous uses in practice and also plays a central role in shaping many visual processing algorithms and systems, as well as their implementation, optimization, and testing. As human being is end receiver of images, one straightforward way for evaluating image quality is subjective testing. The mean opinion score (MOS), subjective quality measurement, has been used for many years. However, it is very expensive and time consuming, which makes it impractical for image processing applications. These drawbacks lead to the development of perceptual image quality assessment (PIQA) metrics that can automatically evaluate the image perceptual quality.

An objective measurement of perceptual quality plays a very important role in many image processing tasks, such as image compression and enhancement. It can be used to dynamically monitor and adjust image quality, optimize algorithms, and benchmark image processing systems [1]. In recent years, a great deal of effort has been made to develop objective image quality metrics that correlate with human visual behaviors in evaluating image quality [1–4].

Depending upon the availability of a “perfect quality” reference image, the image quality assessment (IQA) metrics are classified into full-reference (FR), reduced-reference (RR), and no-reference (NR) [4]. FR metrics are those that need access to an original reference image to produce a quality score that predicts the subjective judgment of a distorted image. NR metrics only require distorted images to predict quality scores. RR metrics are between FR metrics and NR metrics, which require only partial information about the reference image [3]. In this paper, we focus on FR image quality assessment.

Generally, FR metrics measure the distance between a distorted image and its original image in a perceptually meaningful way. FR metrics can also be designed in two ways. One is modeling HVS, which has been regarded as the most appropriate way to measure and predict the perceptual image quality. The underlying assumption is that HVS is sensitive to the differences of visual signals in some respects, such as brightness, contrast, and frequency content. Under this assumption, the strength of the difference between a distorted and its reference image reflects the different perceived sensitivities of HVS. The other way explores signal fidelity criteria that is not based on assumptions about HVS model but is motivated instead by the need to capture the loss of signal structure that HVS hypothetically extracts for cognitive understanding [5].

In this paper, we propose a new framework for PIQA based on TV model in spatial domain. In the proposed PIQA metric, two human visual sensitivity factors, image structures and luminance changes in enclosed regions, are considered. As far as natural image signal is concerned, an evaluation metric needs to consider the characteristics of the image itself, such as image structure and content, to reflect the image visual complexity. On the other hand, from HVS perspective, another important factor to be considered is luminance change of smooth and enclosed regions, as HVS is very sensitive to luminance change. Based on these ideas, we propose a TVPIQA metric in spatial domain, in which we use TV to describe the image structure and the energy of enclosed regions in the difference image to measure luminance changes. Then, TVPIQA is represented by the weighted sum of these two factors.

To successfully assess the image quality, there are two major contributions of the proposed metric. First, we introduce TV model to assess image's structure. The TV comparison between a distorted image and its reference image is applied to measure the distance of an image structural characteristic. Because of the good performance of TV in describing the edges, the proposed TVPIQA metric can describe the image structure information very well. Second, the luminance changes in enclosed regions are also considered in TVPIQA. The energy of enclosed regions in a difference image is used to measure the missing luminance information that is sensitive to human visual system. In addition, in order to make TVPIQA metric closer to perceptual feelings, isolated pixels' energy in difference image is removed based on the idea of just noticeable distortion (JND), and a fast approximation method of calculating difference image's energy is also proposed.

We demonstrate our TVPIQA metric by presenting performance results with extensive subjective databases (Cornell-A57 [6, 7], IVC [8, 9], TID2008 [10, 11], and CSIQ [12, 13]) and comparisons to seven often-used image metrics (PSNR, SSIM [1], IW-PSNR [4], IW-SSIM [4], MS-SSIM [14], VSNR [7], and VIF [5]). Experimental results demonstrate that the performance of TVPIQA metric outperforms other state-of-the-art metrics. It is worth noting that the proposed metric is easy to compute in spatial domain and does not need any other additional information.

The rest of paper is organized as follows. Section 2 presents some related work. The PIQA metric based on TV model is given in Section 3, and the implementation details of TVPIQA metric are also provided in this section. The characteristics of TVPIQA metric are analyzed in Section 4 and its performance is evaluated and discussed in Section 5. We conclude the paper in Section 6.

2. Related Work

According to different methodologies being considered, PIQA metrics can be divided into two categories: HVS features based modeling and signal driven approach [15]. For HVS features based modeling, PIQA metrics are developed based upon systematical modeling of relevant psychophysical properties and physiological knowledge, including temporal/spatial/color decomposition, contrast sensitivity function (CSF), luminance adaptation, and various masking effects [15]. A number of HVS based methods have been proposed in the literature [16–19]. Some have also considered JND model [20, 21]. HVS based methods extrapolate the vision models that have been proposed in the visual psychology literature to PIQA. However, HVS features based methods involve expensive computation and difficulties due to the gap between the knowledge for vision research and the need for engineering modeling [15].

Recently, a lot of research efforts have been concentrated on signal driven PIQA metrics, which are designed from the viewpoint of signal extraction and analysis, such as statistical features, structural distortion, and so forth [1, 4, 22, 23]. Signal driven methods do not attempt to build a comprehensive HVS model regarding quality evaluation. These metrics look at how to represent image features to estimate overall quality, and they often consider psychophysical effects as well, usually based on image content and distortion analysis. However, although some signal fidelity metrics reflect picture quality change, they fail to predict HVS perception because of some problems [15]. For example, not every image change is noticeable and leads to distortion. Therefore, signal driven methods need HVS features to help tackle these problems, so that they can better approximate perceptual quality evaluation.

Variational methods have been extremely successful in wide various fields in image processing and computer vision during last decades [24, 25]. TV model is first introduced by Rudin, Osher, and Fatemi (ROF) in their pioneering work on edge preserving [26]. For an image u, its TV can be formulated by

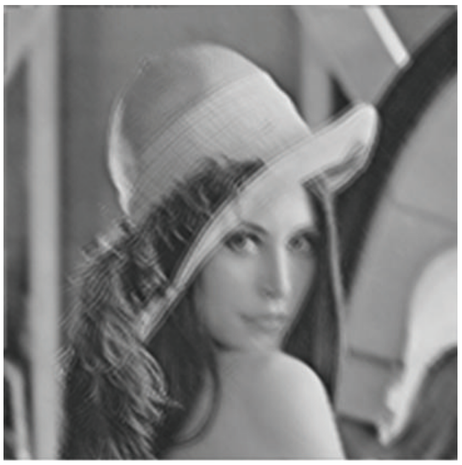
$$\mathrm{TV}(u) = \int_{\Omega} |\nabla u|, \tag{1}$$

where Ω denotes the image domain and ∇ means the gradient operator.

Many research results have shown that the proper norm for an image is the total variation norm, which is essentially L_1 norm of derivative and is more appropriate for image estimation and description in discontinuities [24, 25]. The advantages of TV norm led us to consider using it to measure image structure change, which is the distance between a distorted image and its original image. The proposed TVPIQA metric will focus on two human visual sensitivity factors, that is, image structures and luminance changes in enclosed regions. A significant difference between TVPIQA metric and other PIQA metrics is that TV model is introduced to assess

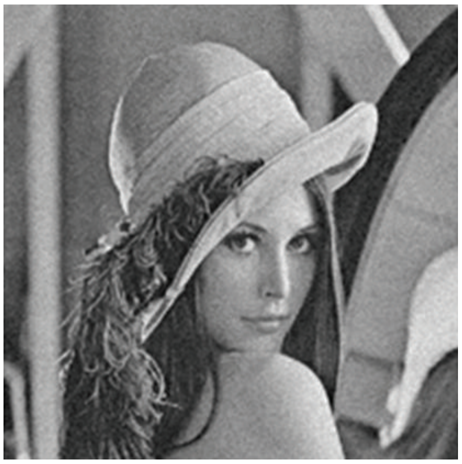
MSE = 0
SSIM = 1
TVPIQA = 1

(a)



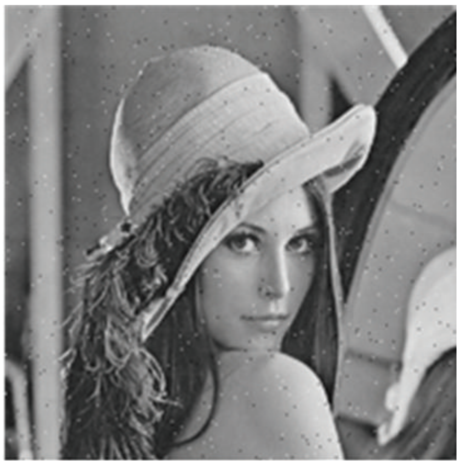
MSE = 66.028
SSIM = 0.875
TVPIQA = 0.863

(b)



MSE = 67.919
SSIM = 0.839
TVPIQA = 0.900

(c)



MSE = 68.862
SSIM = 0.882
TVPIQA = 0.976

(d)

Figure 1: Comparison of "Lena" images with different types of distortions. (a) Original image (512 × 512, 8 bits/pixel). (b) Blurring. (c) Gaussian noise contamination. (d) Impulsive noise contamination.

image structures in spatial domain. Moreover, luminance changes in enclosed regions are also considered.

3. PIQA Metric Based on TV Model

Many signal driven PIQA metrics have been investigated based on the assumption that the loss of perceptual quality is related to the visibility of error signals. Generally, the mean squared error (MSE) or the peak signal-to-noise ratio (PSNR) has been a popular and usual metric to evaluate image quality. But MSE measure exhibits weak performance in assessing perceptual image quality [1]. From Figure 1 where the original "Lena" image is degraded with different distortions, we can see that MSE cannot reflect an image PIQ. The motivation of the paper is to design an appropriate measure for some HVS characteristics, especially image structures and luminance, and develop a novel PIQA metric.

3.1. Framework of Proposed TVPIQA Metric. When a natural image is observed through HVS, the subjective quality measure is affected by many factors. Because human eyes are sensitive to changes of image edges, especially the edge location information, and changes of luminance contrast [27], there are two main factors worthy of attention. One is image edge (structure) information, and the other is luminance information. Therefore, we propose a new TVPIQA metric to measure these two factors and provide a good approximation to perceived image distortion.

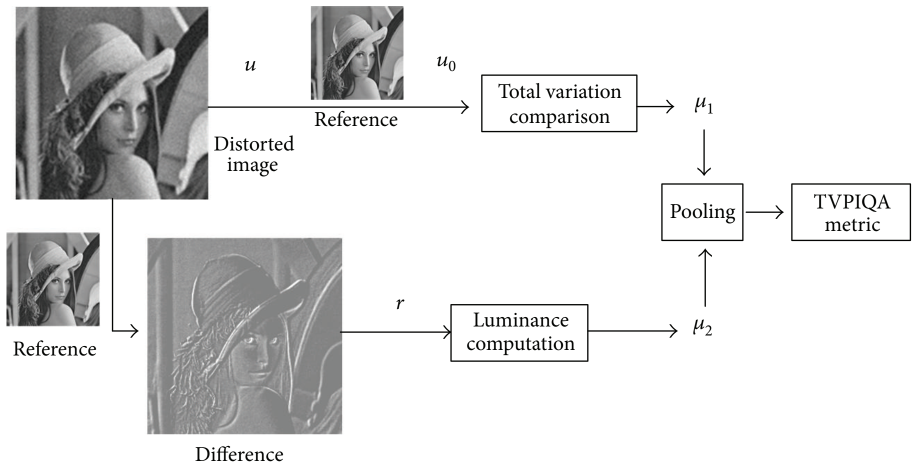


FIGURE 2: Framework of the proposed TVPIQA metric.

Figure 2 shows the framework of the proposed TVPIQA metric, which is separated into two parts: structure part and local region luminance part. In the structure part, the normalized TV comparison between the distorted image and the reference image, denoted by μ_1, is applied to represent the image structure changes. On the other hand, the energy of the enclosed regions in a difference image is used to measure the luminance changes, which is also normalized and denoted by μ_2. Then, the proposed TVPIQA metric is the mean of these two parts as follows:

$$\mathrm{TVPIQA} = \frac{\mu_1 + \mu_2}{2}. \tag{2}$$

3.2. TV Based Structure Measure μ_1. Let u and u_0 represent the degraded image and its reference image, respectively. Because TV norm is very appropriate for image description in discontinuities, the structure's change is measured by the TV difference between a reference image and its degraded image

$$\mathrm{TV}_{\mathrm{struct}} = \left\| \mathrm{TV}(u) - \mathrm{TV}(u_0) \right\|_1, \tag{3}$$

where $\| \cdot \|_1$ represents L_1 norm and $\mathrm{TV}(u)$ denotes the total variation of the image u, expressed in discrete form

$$\mathrm{TV}(u) = \sum_{(i,j)\in\Omega} \left(\sqrt{\left(u_{i,j} - u_{i+1,j}\right)^2 + \left(u_{i,j} - u_{i,j+1}\right)^2} \right), \tag{4}$$

where $u_{i,j}$ represents the intensity value at pixel (i, j).

Although the above measurement can work to assess the structure change, it is not a normalized measure and cannot be used as an evaluation to describe subjective feelings about image's quality. Therefore, according to $\|\mathrm{TV}(u) - \mathrm{TV}(u_0)\|_1^2 \geq 0$, the normalized perceptual distance for image structure is derived and defined by

$$\begin{aligned}\mu_1 = \frac{1}{N} \sum \Bigg(\Bigg(& 2\sqrt{\left(u_{i,j} - u_{i+1,j}\right)^2 + \left(u_{i,j} - u_{i,j+1}\right)^2} \\ & \times \sqrt{\left(u_{0i,j} - u_{0i+1,j}\right)^2 + \left(u_{0i,j} - u_{0i,j+1}\right)^2} + c \Bigg) \\ & \times \Big(\left(u_{i,j} - u_{i+1,j}\right)^2 + \left(u_{i,j} - u_{i,j+1}\right)^2 \\ & + \left(u_{0i,j} - u_{0i+1,j}\right)^2 + \left(u_{0i,j} - u_{0i,j+1}\right)^2 + c \Big)^{-1} \Bigg),\end{aligned} \tag{5}$$

where N is the image size and c is a constant and set to 75 according to our experiments. It is obvious that $\mu_1 \in (0, 1]$.

3.3. Local Region Luminance Measure μ_2. A difference image represents the information loss in a distorted image and is defined by the difference between a reference image and its distorted image; that is, $r = u_0 - u$. Because HVS is also sensitive to luminance changes when observing an image, the energy of a difference image is used to measure the missing luminance information. Furthermore, based on the idea of JND modeling, that is, not every change in an image is noticeable [15], the isolated pixels' energy in a difference image is filtered out. The energy measure of a difference image E_r, which measures the luminance loss of a distorted image, is defined by

$$E_r = \frac{1}{N} \sum_{(i,j)\in\Omega'} r_{i,j}^2, \tag{6}$$

where Ω' represents the enclosed regions in a difference image and $r_{i,j}$ is the intensity value at pixel (i, j) in difference image. To compute the energy measure E_r efficiently and

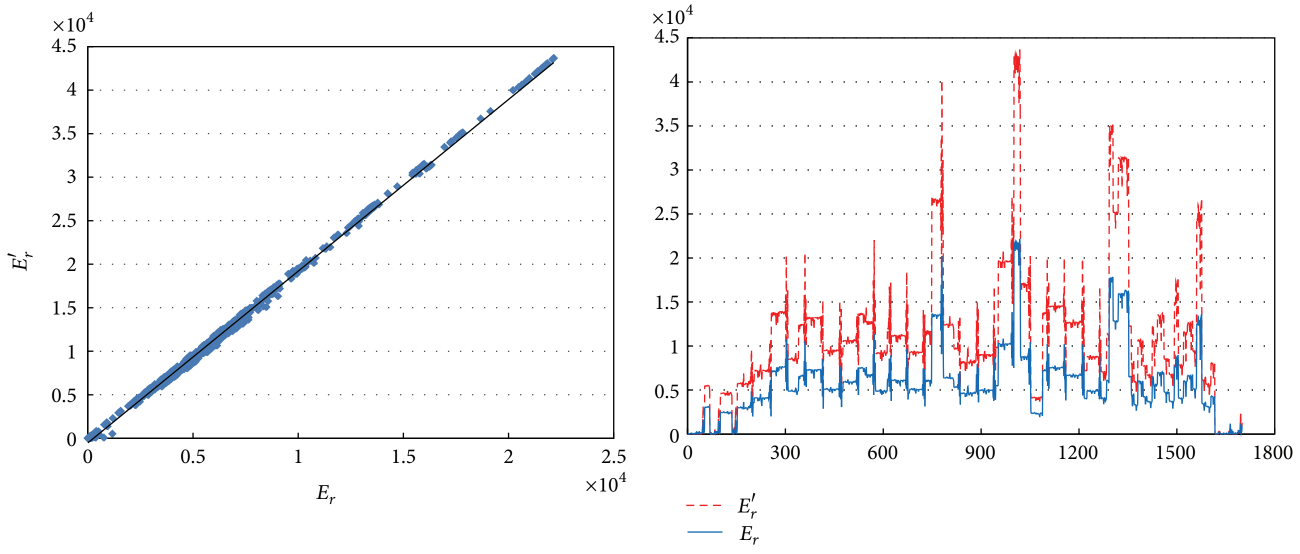


FIGURE 3: Comparison of E_r and E'_r for TID2008. (a) E'_r-E_r curve. The correlation coefficient value between E'_r and E_r is 0.9988. (b) E_r and E'_r curves for different distorted images.

avoid the judgment of the enclosed regions, we proposed an approximation of the difference image's energy E'_r defined as

$$E'_r = \frac{1}{N} \sum_{(i,j)\in\Omega} \left(r_{i,j} r_{i+1,j} + r_{i,j} r_{i,j+1} \right), \quad (7)$$

where Ω represents the whole regions in a difference image.

To test the relationship between E_r and E'_r, we applied these two functions to TID2008 image database [10, 11], which includes 1700 distorted images generated from 25 reference images with 17 distortion types at four distortion levels. Figure 3(a) shows the relationship between E'_r and E_r. The value of correlation coefficient is 0.9988, indicating that E'_r is highly related to E_r. In Figure 3(b), the horizontal axis indicates the image number and the vertical axis indicates the energy value. Figure 3(b) shows the change curves of E'_r and E_r plotted by all distorted images in TID2008. From Figure 3(b), we can see that E'_r and E_r almost have the same change trend. Therefore, we can use E'_r, instead of E_r, to measure the energy of a difference image.

Considering that human perception is more sensitive to luminance contrast rather than to absolute luminance, we adjust E'_r according to the mean intensity of a difference image

$$E'_r = \frac{1}{N} \sum_{(i,j)\in\Omega} \left(\left(r_{i,j} - \overline{r}\right)\left(r_{i+1,j} - \overline{r}\right) + \left(r_{i,j} - \overline{r}\right)\left(r_{i,j+1} - \overline{r}\right) \right), \quad (8)$$

where $\overline{r}$ represents the mean intensity of a difference image; that is, $\overline{r} = (1/N)\sum_{(i,j)\in\Omega} r_{i,j}$.

In order to obtain the normalized measure for luminance change, we need to find the maximum difference image's energy according to the reference image. In the case of consistency of the overall image energy, we assume that the distorted image, in which intensity values in all pixels are the same and equal to the mean intensity of the reference image, corresponds to the maximum luminance change. Based on this assumption, the maximum difference image is denoted by $r_{\max} = u_0 - \overline{u_0}$. It describes the maximum loss of luminance information in an original reference image. Then, the normalized perceptual distance measure for image luminance change is defined by

$$\mu_2 = 1 - \sqrt{\frac{E'_r}{E'_{r_{\max}}}}, \quad (9)$$

where $E'_{r_{\max}}$, computed by (8), represents the energy measure of the maximum difference image $r_{\max}$. Obviously, $\mu_2 \in (0, 1]$. The higher the value of μ_2 is, the less luminance information lose.

4. Analysis of the Proposed TVPIQA Measure

This section analyzes some properties of the proposed TVPIQA measure, such as symmetry, boundedness, and unique maximum. Meanwhile, the difference of TVPIQA measure is also discussed in evaluating different types (contrast change, noise contamination, and blurring) of distorted images in this section.

4.1. Properties of TVPIQA Measure

(1) *Symmetry*. Because the structure measure function $\mu_1(x, y)$ is derived by $\|\mathrm{TV}(u) - \mathrm{TV}(u_0)\|_1^2 \geq 0$, $\mu_1(x, y) = \mu_1(y, x)$ is obvious. On the other hand, the energy function satisfies the symmetry; that is, $E'_r(x, y) = E'_r(y, x)$, and the energy

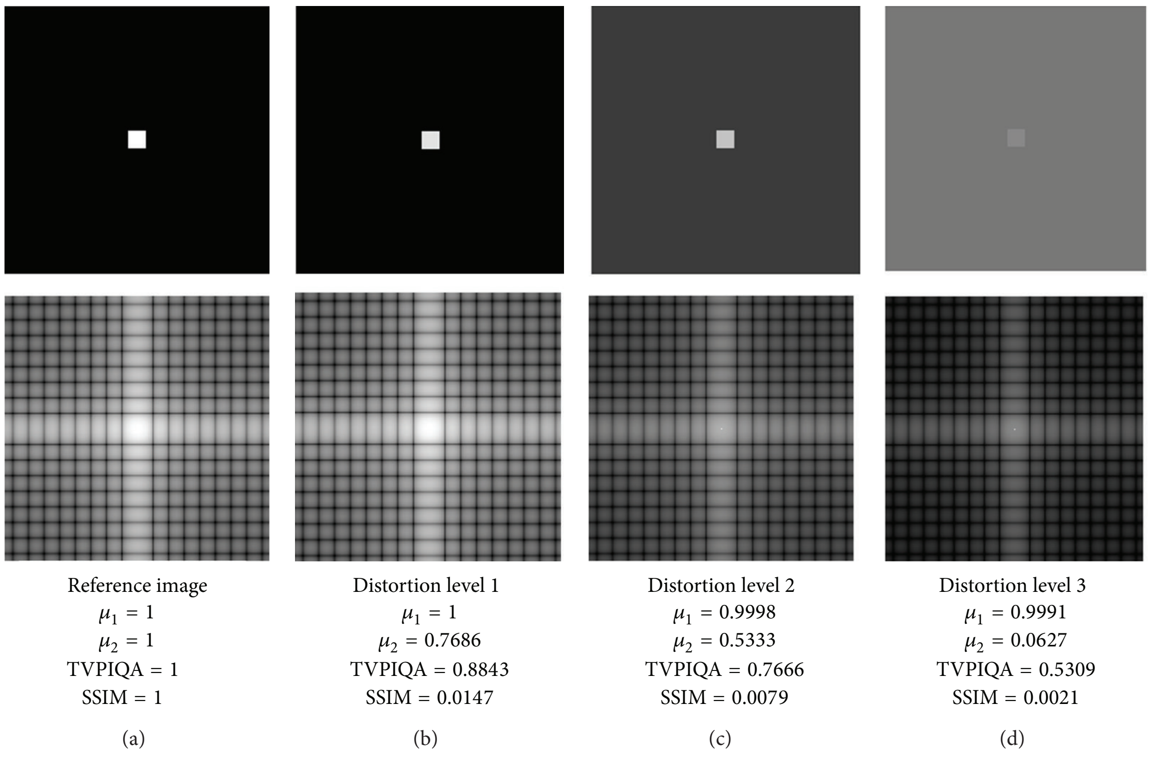


FIGURE 4: Contrast change distortion analysis in frequency domain. The first row images are the reference image and contrast change distorted images. The second row images are Fourier spectrums of the first row images. (a) Reference image. Background is black with a white square. The gray values of the background and the square are 0 and 255, respectively. (b) Contrast change distortion level 1. Background is 30, and square is 226. (c) Contrast change distortion level 2. Background is 60, and square is 196. (d) Contrast change distortion level 3. Background is 120, and square is 136.

measure function $\mu_2(x, y) = 1 - \sqrt{E'_r(x, y)/E'_{r_{\max}}}$ also satisfies the symmetry. Therefore, the proposed TVPIQA measure, $\mathrm{TVPIQA}(x, y) = (\mu_1(x, y) + \mu_2(x, y))/2$, is symmetric.

(2) *Boundedness and Unique Maximum*. According to the definition of $\mu_1(x, y)$ and $\mu_2(x, y)$ in Section 3, measure $\mu_1(x, y) \in (0, 1]$ and measure $\mu_2(x, y) \in (0, 1]$. So, TVPIQA metric is bounded; that is, $\mathrm{TVPIQA}(x, y) \in (0, 1]$. Only when a distorted image is the same as its reference image, the structure measure $\mu_1(x, y) = 1$ and energy measure $\mu_2(x, y) = 1$; that is, if and only if $x = y$, $\mathrm{TVPIQA}(x, y) = 1$.

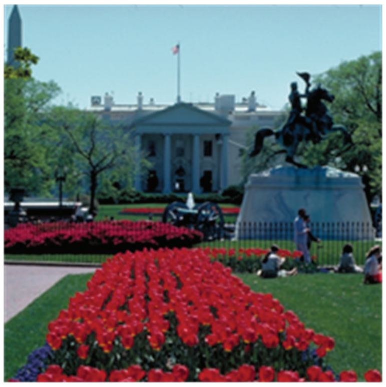
4.2. TVPIQA Measure for Different Types of Distortions. Due to the limited considered factors in designing PIQA metric and limited understanding of HVS, it is impossible for any PIQA metric to measure all kinds of distortions in the same measurement scale. To some distortions, a PIQA metric may be gentle, while being critical to other distortions. In our extensive experiments, the proposed TVPIQA metric is more critical in evaluating contrast distortions, compared with evaluating other distortions. This performance of TVPIQA exactly reflects HVS contrast sensitivity characteristic.

Figure 4 shows different levels of contrast distortions for an artificial image. In Figure 4, μ_1 measures structure changes in the distorted images, and μ_2 measures luminance changes. TVPIQA and SSIM measure the distances from distorted images to the reference image. From Figure 4, we can see that structure measure in the distorted images, μ_1, changes extremely slowly. However, the energy measure in the distorted images, μ_2, declines rapidly. The Fourier spectrums of the distorted images also show the same changes. Figure 5 shows another example, and the test image is “1600” images in CSIQ database. The same situation can be observed from Figures 5(b), 5(c), and 5(d) that μ_1 changes slowly and μ_2 drops dramatically. For the same distortion level with different distortion types, shown in Figures 5(d), 5(e), and 5(f), the contrast distortion has the largest drop between luminance measure and structure measure.

According to Figures 4 and 5, TVPIQA measure shows different measure scales for contrast distortions and other distortions, such as noise contamination and blurring. However, this does not mean that the performance of TVPIQA measure is not good, because, for HVS perception, not every change yields the same extent of perceptual effect with the same magnitude of change [15]. Therefore, to evaluate TVPIQA measure performance, the contrast distortion will be discussed separately in the next section.

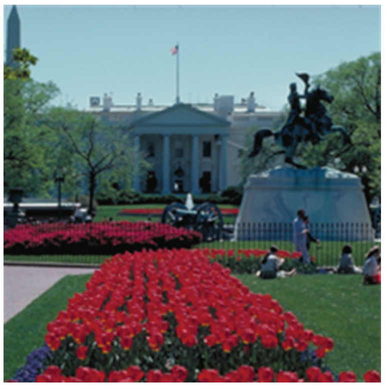
5. Experimental Results

In this section, we validate the performance of the proposed TVPIQA measure and compare it with other seven IQA measures, that is, PSNR, SSIM [1], IW-PSNR [4], IW-SSIM



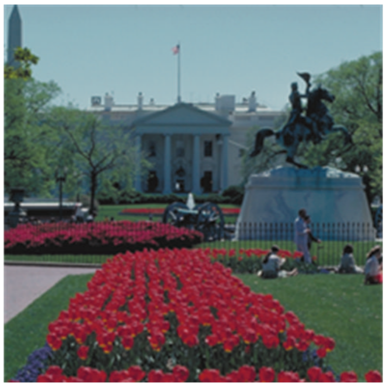
Reference image
$\mu_1 = 1$
$\mu_2 = 1$
TVPIQA = 1
SSIM = 1

(a)



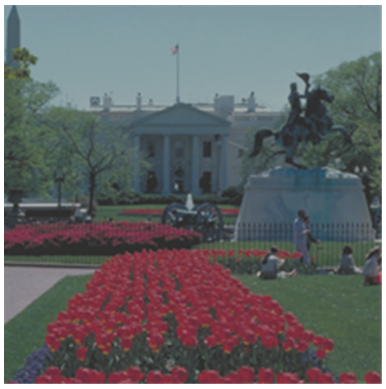
Contrast level 1
$\mu_1 = 0.9936$
$\mu_2 = 0.8903$
TVPIQA = 0.9419
SSIM = 0.9842

(b)



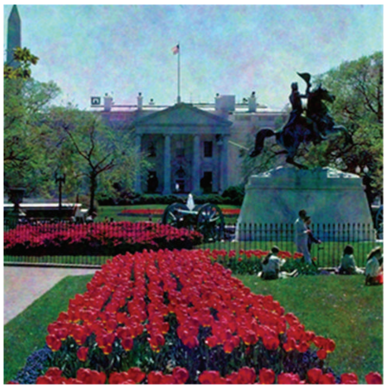
Contrast level 2
$\mu_1 = 0.9744$
$\mu_2 = 0.7498$
TVPIQA = 0.8621
SSIM = 0.9439

(c)



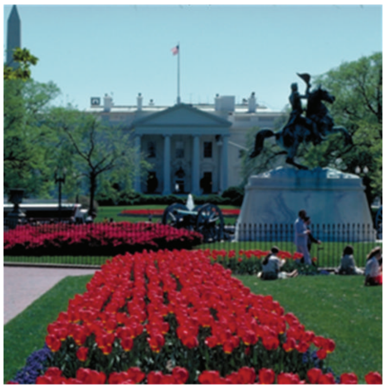
Contrast level 3
$\mu_1 = 0.8984$
$\mu_2 = 0.5233$
TVPIQA = 0.7109
SSIM = 0.8260

(d)



Pink noise level 3
$\mu_1 = 0.9365$
$\mu_2 = 0.9057$
TVPIQA = 0.9211
SSIM = 0.9334

(e)



Blurring level 3
$\mu_1 = 0.9164$
$\mu_2 = 0.9666$
TVPIQA = 0.9415
SSIM = 0.9458

(f)

FIGURE 5: TVPIQA measure comparison for different type distortions. The test image is the "1600" images in CSIQ database. (a) Reference image. (b) Contrast change distortion level 1. (c) Contrast change distortion level 2. (d) Contrast change distortion level 3. (e) Additive Gaussian pink noise contamination level 3. (f) Blurring level 3.

[4], MS-SSIM [14], VSNR [7], and VIF [5]. PSNR is widely used in the image processing field and is also a useful baseline comparison. SSIM, MS-SSIM, visual signal-to-noise ratio (VSNR), and VIF are state-of-the-art measures that have demonstrated competitive performance. The information content weighted PSNR (IW-PSNR) and information content weighted SSIM (IW-SSIM) measures have been confirmed having the best overall performance compared with the previous measures. So, IW-PSNR and IW-SSIM are also good benchmarks to evaluate the new TVPIQA measure.

The proposed TVPIQA measure and other seven measures are evaluated on four publicly available subjective image databases that are widely recognized in the IQA research community, that is, Cornell-A57 [6, 7], IVC [8, 9], TID2008 [10, 11], and CSIQ [12, 13]. Two different types of subjective quality scores have been used: MOS and differential MOS (DMOS) in these image databases.

The Cornell-A57 database [6, 7] was created at Cornell University. It contains 54 distorted images with six types of distortions including quantization distortion, noise contamination, and blurring. The IVC database [8, 9] includes 185 distorted images generated from ten original images. There are four types of distortions that are (a) JPEG compression, (b) JPEG2000 compression, (c) local adaptive resolution (LAR) coding, and (d) blurring. The Tampere Image Database 2008 (TID2008), introduced in the previous section, is intended to evaluate full-reference image visual quality assessment metrics. It has 17 types of distortions, such as noise distortion, blur distortion, and contrast change. The categorical image quality (CSIQ) database [12, 13] was developed at Oklahoma

State University. It consists of 30 original images and 866 distorted images using six different types of distortions at four to five different levels of distortion. The distortion types include JPEG compression, JPEG2000 compression, global contrast decrements, additive pink Gaussian noise, and Gaussian blurring.

The performance of any objective visual quality assessment metric is evaluated by measuring its correlation with human perception. In order to do so, the objective quality scores of IQA metrics are correlated with subjective scores using a variety of statistical measures such as the correlation coefficient (CC), Spearman's rank correlation coefficient (SRCC), and Kendall's rank correlation coefficient (KRCC). To compare performance of different IQA measures, three evaluation metrics, that is, CC, SRCC, and KRCC, are used in experiments as shown in the following.

Correlation coefficients evaluate the prediction accuracy and measure linear dependence between the subjective and the objective scores. CC is defined as

$$\mathrm{CC} = \frac{\sum_{i=1}^{N} (s_i - \bar{s}) \cdot (o_i - \bar{o})}{\sqrt{\sum_{i=1}^{N} (s_i - \bar{s})^2} \sqrt{\sum_{i=1}^{N} (o_i - \bar{o})^2}}, \tag{10}$$

where $\bar{s}$ is the mean value of subjective scores s_i, $i = 1, \ldots, N$ and $\bar{o}$ is the mean value of objective scores o_i.

Spearman's rank correlation coefficients measure the prediction of monotonicity [28]. SRCC is given by

$$\begin{gathered} \mathrm{SRCC} = 1 - \frac{6 \sum_{i=1}^{N} d_i^2}{N (N^2 - 1)}, \\ d_i = R_{s_i} - R_{o_i}, \end{gathered} \tag{11}$$

where R_{s_i} and R_{o_i} are the ith image's ranks in subjective and objective evaluations, respectively. SRCC is a nonparametric rank-based correlation metric, independent of any monotonic nonlinear mapping between subjective and objective scores [4].

Kendall's rank correlation coefficient (KRCC) is another nonparametric rank correlation metric computed by

$$\mathrm{KRCC} = \frac{N_c - N_d}{N (N - 1) / 2}, \tag{12}$$

where N_c and N_d are the numbers of concordant and discordant pairs in the data set, respectively.

According to the above definitions, larger CC, SRCC, and KRCC values, close to 1, indicate that the objective and subjective scores correlate better, that is to say, a better performance of IQA metric. In our performance comparisons, CC, SRCC, and KRCC of seven metrics are either computed or referenced from some research works [29–31]. Among these metrics, PSNR, IW-PSNR, SSIM, MS-SSIM, and IW-SSIM are referred to Zhou Wang's research [29], while VSNR and VIF are referred to Lin Zhang's research [30, 31].

Table 1 shows our test results of eight IQA metrics using four databases. As analyzed in Section 4, the contrast distorted images in TID2008 and CSIQ databases are discussed separately. For each evaluation metric in each test, we

Table 1: Performance comparisons of eight IQA metrics on four publicly available image databases.

Metric	CC	SRCC	KRCC
Cornell-A57 database (54 images) [6, 7]			
PSNR	0.6346	0.6189	0.4309
SSIM [1]	0.7531	0.8066	0.6058
TVPIQA	**0.8797**	**0.8328**	**0.6510**
IW-PSNR [4]	0.8752	0.8759	0.6967
IW-SSIM [4]	0.8918	0.8709	0.6842
MS-SSIM [14]	0.8396	0.8414	0.6478
VSNR [7]	**0.9147**	**0.9355**	**0.8031**
VIF [5]	0.6141	0.6223	0.4589
IVC database (185 images) [8, 9]			
PSNR	0.6705	0.6884	0.5218
SSIM [1]	0.8092	0.9018	0.7223
TVPIQA	**0.8551**	**0.9043**	**0.7242**
IW-PSNR [4]	0.8900	0.8998	0.7165
IW-SSIM [4]	**0.7943**	**0.9125**	**0.7339**
MS-SSIM [14]	0.7854	0.8980	0.7203
VSNR [7]	0.7824	0.7993	0.6053
VIF [5]	0.8800	0.8964	0.7158
TID2008 database without contrast distortion (1600 images) [10, 11]			
PSNR	0.5599	0.5974	0.4394
SSIM [1]	0.7519	0.7912	0.5912
TVPIQA	**0.8592**	**0.8772**	**0.6869**
IW-PSNR [4]	0.6353	0.7855	0.6074
IW-SSIM [4]	**0.8217**	**0.8748**	**0.6868**
MS-SSIM [14]	0.8031	0.8742	0.6798
VSNR [7]	0.3238	0.7992	0.6096
VIF [5]	0.7622	0.7383	0.5777
TID2008 database with only contrast distortion (100 images) [10, 11]			
PSNR	0.5770	0.5830	0.4178
SSIM [1]	0.5061	0.5204	0.3891
TVPIQA	**0.6210**	**0.6054**	**0.4421**
IW-PSNR [4]	0.5691	0.5627	0.4000
IW-SSIM [4]	0.7605	0.6251	0.4675
MS-SSIM [14]	0.7556	0.6379	0.4793
VSNR [7]	0.4048	0.4090	0.2686
VIF [5]	**0.8706**	**0.8094**	**0.5678**
CSIQ database without contrast distortion (750 images) [12, 13]			
PSNR	0.8544	0.9061	0.7237
SSIM [1]	0.8475	0.9247	0.7534
TVPIQA	**0.9040**	**0.9583**	**0.8170**
IW-PSNR [4]	0.9267	0.9523	0.8069
IW-SSIM [4]	**0.8292**	**0.9544**	**0.8096**
MS-SSIM [14]	0.8481	0.9506	0.7984
VSNR [7]	0.8471	0.9266	0.7575
VIF [5]	0.9137	0.9181	0.7561

Table 1: Continued.

Metric	CC	SRCC	KRCC
CSIQ database with only contrast distortion (116 images) [12, 13]			
PSNR	0.8887	0.8621	0.6449
SSIM [1]	0.7666	0.7922	0.5779
TVPIQA	**0.9491**	**0.9543**	**0.8162**
IW-PSNR [4]	0.9125	0.9230	0.7508
IW-SSIM [4]	**0.9098**	**0.9539**	**0.8168**
MS-SSIM [14]	0.9003	0.9526	0.8123
VSNR [7]	0.8680	0.8795	0.7022
VIF [5]	0.9336	0.9404	0.7901

Table 2: Average performance over four databases.

Metric	CC	SRCC	KRCC
Database size-weighted average without contrast distortion			
PSNR	0.6547	0.6938	0.5275
SSIM [1]	0.7837	0.8381	0.6479
TVPIQA	**0.8723**	**0.9017**	**0.7265**
IW-PSNR [4]	0.7429	0.8439	0.6748
IW-SSIM [4]	**0.8234**	**0.9005**	**0.7257**
MS-SSIM [14]	0.8156	0.8973	0.7164
VSNR [7]	0.5205	0.8389	0.6562
VIF [5]	0.8114	0.7993	0.6368
Database size-weighted average with all distorted images			
PSNR	0.6616	0.6968	0.5284
SSIM [1]	0.7731	0.8249	0.6358
TVPIQA	**0.8666**	**0.8933**	**0.7201**
IW-PSNR [4]	0.7437	0.8371	0.6682
IW-SSIM [4]	**0.8247**	**0.8929**	**0.7203**
MS-SSIM [14]	0.8170	0.8904	0.7119
VSNR [7]	0.5307	0.8253	0.6442
VIF [5]	0.8186	0.8055	0.6406

highlight our proposed TVPIQA metric and the best of other seven metrics with boldface.

From the experimental results in Table 1, we have two major observations. TVPIQA metric has similar performance to IW-SSIM on Cornell-A57 and IVC databases. Without considering the contrast distortion, the proposed TVPIQA metric has the best performance on TID2008 and CSIQ databases. On the other hand, TVPIQA metric has the highest SRCC value in evaluating the contrast distortion in CSIQ database. However, many metrics do not work well on the contrast distortion in TID2008 database. The main reason is that contrast enhancement images in TID2008 have high subjective scores for HVS perception, while they are measured as the distortion by some objective evaluation metrics.

To evaluate the overall performance of IQA metrics under comparison, Table 2 presents the average CC, SRCC, and KRCC results over four databases, where the average values are computed in two cases. In the first case, without considering the contrast distortion, the correlation scores are computed by the size weighted average method. Different weights are given to different databases, depending upon their sizes (measured as the numbers of images, i.e., 54 for Cornell-A57, 185 for IVC, 1600 for TID2008, and 750 for CSIQ databases), while in the second case, the contrast distortions in TID2008 and CSIQ are also considered, and the correlation scores are still averaged according to the numbers of images, 100 for TID2008 contrast distortion and 116 for CSIQ contrast distortion.

From Table 2, it can be observed that our proposed TVPIQA metric has better overall performance than other IQA metrics. Although IW-SSIM and TVPIQA nearly have the same performance from test results, it is worth mentioning that, from the view point of computation complexity, TVPIQA metric achieves this excellent performance only by computing the image structure and energy information in spatial domain, which does not need any preprocessing, such as image transform operation, extra image analysis operation, and so forth. However, IW-SSIM needs more computation time to information content weights.

6. Conclusions

In this paper, we propose a new framework for TV based perceptual image quality assessment measure in spatial domain. The proposed TVPIQA measure focuses on two human visual sensitivity factors, image structures and luminance changes. A significant difference between TVPIQA measure and other IQA measures is that TV model is introduced to assess image structures. Meanwhile, the energy of enclosed regions in a difference image is used to measure the missing luminance information which is also sensitive to human visual system. Extensive experimental results with four publicly available independent image databases demonstrate that the proposed TVPIQA measure achieves the best overall performance when compared with other seven popular IQA measures.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

This work is partially supported by National Natural Science Foundation of China (Grant no. 61303127), Western Light Talent Culture Project of Chinese Academy of Sciences (Grant no. 13ZS0106), Project of Science and Technology Department of Sichuan Province (Grant nos. 2011JQ0041 and 11ZS2009), and Key Program of Education Department of Sichuan Province (Grant nos. 11ZA130 and 13ZA0169).

References

[1] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: from error visibility to structural similarity," *IEEE Transactions on Image Processing*, vol. 13, no. 4, pp. 600–612, 2004.

[2] "Perceptual criteria for image quality evaluation," in *Handbook of Image and Video Processing*, T. N. Pappas, R. J. Safranek, J. Chen, and A. Bovik, Eds., Academic Press, New York, NY, USA, 2nd edition, 2005.

[3] Z. Wang and A. C. Bovik, *Modern Image Quality Assessment*, Morgan and Claypool Publishers, San Rafael, Calif, USA, 2006.

[4] Z. Wang and Q. Li, "Information content weighting for perceptual image quality assessment," *IEEE Transactions on Image Processing*, vol. 20, no. 5, pp. 1185–1198, 2011.

[5] H. R. Sheikh and A. C. Bovik, "Image information and visual quality," *IEEE Transactions on Image Processing*, vol. 15, no. 2, pp. 430–444, 2006.

[6] D. M. Chandler and S. S. Hemami, "Cornell-A57 Database," http://foulard.ece.cornell.edu/dmc27/vsnr/vsnr.html.

[7] D. M. Chandler and S. S. Hemami, "VSNR: a wavelet-based visual signal-to-noise ratio for natural images," *IEEE Transactions on Image Processing*, vol. 16, no. 9, pp. 2284–2298, 2007.

[8] A. Ninassi, P. Le Callet, and F. Autrusseau, "Pseudo no reference image quality metric using perceptual data hiding," in *Human Vision and Electronic Imaging*, vol. 6057 of *Proceedings of the SPIE*, San Jose, Calif, USA, January 2006.

[9] A. Ninassi, P. Le Callet, and F. Autrusseau, "Subjective quality assessment: IVC database," http://www2.irccyn.ec-nantes.fr/ivcdb.

[10] N. Ponomarenko, F. Battisti, K. Egiazarian, J. Astola, and V. Lukin, "Metrics performance comparison for color image database," in *4th International Workshop on Video Processing and Quality Metrics*, Scottsdale, Ariz, USA, January 2009.

[11] N. Ponomarenko and K. Egiazarian, "TAMPERE IMAGE DATABASE 2008 TID2008, version 1.0," http://www.ponomarenko.info/tid2008.htm.

[12] E. C. Larson and D. M. Chandler, "Categorical image quality (CSIQ) database," http://vision.okstate.edu/csiq.

[13] E. C. Larson and D. M. Chandler, "Most apparent distortion: full-reference image quality assessment and the role of strategy," *Journal of Electronic Imaging*, vol. 19, no. 1, Article ID 011006, 2010.

[14] Z. Wang, E. P. Simoncelli, and A. C. Bovik, "Multi-scale structural similarity for image quality assessment," in *Proceedings of the Thirty-Seventh Asilomar Conference on Signals, Systems and Computers*, pp. 1398–1402, Pacific Grove, Calif, USA, November 2003.

[15] W. Lin and C.-C. Jay Kuo, "Perceptual visual quality metrics: a survey," *Journal of Visual Communication and Image Representation*, vol. 22, no. 4, pp. 297–312, 2011.

[16] S. Daly, "The visible difference predictor: an algorithm for the assessment of image fidelity," in *Human Vision, Visual Processing, and Digital Display*, vol. 1616 of *Proceedings of SPIE*, pp. 2–15, 1992.

[17] O. D. Faugeras, "Digital color image processing within the framework of a human visual model," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 27, no. 4, pp. 380–393, 1979.

[18] F. X. Lukas and Z. L. Budrikis, "Picture quality prediction based on a visual model," *IEEE Transactions on Communications*, vol. 30, no. 7, pp. 1679–1692, 1982.

[19] A. B. Watson, "DCTune: a technique for visual optimization of DCT quantization matrices for individual images," *Society for Information Display Digest of Technical Papers*, vol. 24, pp. 946–949, 1993.

[20] L. Ma and K. N. Ngan, "Adaptive block-size transform based just-noticeable difference profile for images," in *Proceedings of the 10th Pacific Rim Conference on Multimedia*, 2009.

[21] W. Lin, L. Dong, and P. Xue, "Visual distortion gauge based on discrimination of noticeable contrast changes," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 15, no. 7, pp. 900–909, 2005.

[22] M. Miyahara, K. Kotani, and V. Ralph Algazi, "Objective picture quality scale (PQS) for image coding," *IEEE Transactions on Communications*, vol. 46, no. 9, pp. 1215–1226, 1998.

[23] I. Avcibaş, B. Sankur, and K. Sayood, "Statistical evaluation of image quality measures," *Journal of Electronic Imaging*, vol. 11, no. 2, pp. 206–223, 2002.

[24] T. F. Chan, S. Esedoglu, F. Park, and A. Yip, "Recent developments in total variation image restoration," in *Handbook of Mathematical Models in Computer Vision*, Springer, 2005.

[25] T. F. Chan, J. Shen, and L. Vese, "Variational PDE models in image processing," *Notices of the American Mathematical Society*, vol. 50, no. 1, pp. 14–26, 2003.

[26] L. I. Rudin, S. Osher, and E. Fatemi, "Nonlinear total variation based noise removal algorithms," *Physica D: Nonlinear Phenomena*, vol. 60, no. 1–4, pp. 259–268, 1992.

[27] T. T. Norton, D. A. Corliss, and J. E. Bailey, *Psychophysical Measurement of Visual Function*, Butterworth-Heinemann, Boston, Mass, USA, 2002.

[28] VQEG, "Final report from the video quality experts group on the validation of objective models of video quality assessment," 2000, http://www.vqeg.org/.

[29] "IW-SSIM: Information Content Weighted Structural Similarity Index for Image Quality Assessment," 2011, https://ece.uwaterloo.ca/~z70wang/research/iwssim/.

[30] "Evaluation of VIF," 2011, http://sse.tongji.edu.cn/linzhang/IQA/Evalution_VIF/eva-VIF.htm.

[31] "Evaluation of VSNR," 2011, http://sse.tongji.edu.cn/linzhang/IQA/Evalution_VSNR/eva-VSNR.htm.

Hindawi Publishing Corporation
Journal of Applied Mathematics
Volume 2014, Article ID 943094, 8 pages
http://dx.doi.org/10.1155/2014/943094

Research Article

Modeling a Heterogeneous Embedded System in Coloured Petri Nets

Huafeng Zhang,[1] Hehua Zhang,[2] Ming Gu,[2] and Jiaguang Sun[2]

[1] *School of Computer Science, TNList, Tsinghua University, Beijing 100084, China*
[2] *School of Software, TNList, Tsinghua University, Beijing 100084, China*

Correspondence should be addressed to Huafeng Zhang; ron.huafeng@gmail.com

Received 4 February 2014; Accepted 28 February 2014; Published 31 March 2014

Academic Editor: X. Song

Embedded devices are everywhere now and, unlike personal computers, their systems differ in implementation languages and behaviors. Interactions of different devices require programmers to master programming paradigms in all related languages. So, a defect may occur if differences in systems' behaviors are ignored. In this paper, a heterogeneous system which is composed of two subsystems is introduced and we point out a potential defect in this system caused by an interface mismatch. Then, a state based approach is applied to verify our analysis of the system.

1. Introduction

Embedded devices are everywhere from factories to living rooms and embedded development becomes a new trend in current startup teams. According to the physical carrier of their system, traditional embedded devices can be divided into different domains such as DSP, FPGA, ARM, and PLC. Different carriers mean different programming languages and styles which relate closely to the behavior of the system on the device. For example, the VHDL programming language is widely used in the embedded system's implementation on FPGA board and the common behavior of such systems is synchronous reactive and dataflow oriented [1]. An embedded system on ARM board mostly adopts the C programming language which exploits the power of interrupts and threads asynchronously [2]. There are good engineering practices to follow for developers in such application domains. However, as the rapid evolution of embedded devices, embedded systems confront more and more complex usage scenarios with new problems to be resolved [3]. One of these problems is incompatible programming paradigms between heterogeneous developing techniques which today's embedded systems utilize [4]. A typical embedded-C programmer does not know much about how to write a piece of good VHDL program although he may learn VHDL's grammar somehow and neither does a VHDL programmer know much about the embedded-C stuff. The lack of knowledge in other's programming domain leads to misunderstandings of assumptions and guarantees requirements between interfaces of heterogeneous embedded systems. A trend in handling this heterogeneous development work flow is unifying the programming language used by FPGA, ARM, and DSP developers [5], but it takes efforts to learn a brand-new language even for an experienced developer. Another approach is model based development [6–8]. The developer builds a unified model for the whole system and automatically generates dedicated code for each part according to their specific programming paradigm. The model can be analyzed and verified to guarantee its reliability and enhance its performance before the code generation process [9, 10].

Serious defects between interactions may be undiscovered until on-board tests. The longer the defects go unnoticed, the more time will be spent on them. There are some testing based methods targeting this problem [11]. However, they are not sufficient to guarantee the system's correctness. Analysis based methods [12, 13] can also alleviate the pain in embedded system development.

In this paper, we investigate a case of a heterogeneous embedded system in a real engineering practice and build a model for the system using Colored Petri Nets (CPN) [14] modeling language. Then, we reveal how a normal subsystem on ARM board and a normal subsystem on FPGA board together lead to a defective system step by step. Then, the model checking approach will be applied on the model to verify our analysis.

2. Preliminaries

2.1. Petri Nets. Petri nets [15] are a simple and expressive modeling formalism, which allows users to model complex systems in various paradigms.

Petri nets are a tuple of $\langle P, T, A\rangle$, where $P = p_1, p_2, \ldots, p_n$ is a finite set of places, $T = t_1, t_2, \ldots, t_n$ is a finite set of transitions, and $A = a_1, a_2, \ldots, a_n$ is a finite set of arcs, while P, T, and A are pairwise disjoint. Each place contains a set of markers called tokens and the number of tokens in each place can be 0, 1, or more. The distribution of tokens in all places is called the marking of the Petri net which is denoted by μ. The marking μ can be viewed as an n-vector, $\mu = (\mu_1, \mu_2, \ldots, \mu_n)$, where $n = |P|$ and each μ_i indicates the number of tokens in their corresponding place $p_i, i = 1, \ldots, n$. The marking of a Petri net defines the net's state and it can only be changed by a transition t which moves tokens from a set of places $I(t)$ to another set of places $O(t)$, where $I(t) \subset P$, $O(t) \subset P$, and $t \in T$. The transition t's transfer of tokens is called firing. When t is fired, it removes a token from each place of $p_{t\_\text{in}_1}, p_{t\_\text{in}_2}, \ldots, p_{t\_\text{in}_n}$ in $I(t)$ and adds a token to each place of $p_{t\_\text{out}_1}, p_{t\_\text{out}_2}, \ldots, p_{t\_\text{out}_m}$ in $O(t)$, where $n = |I(t)|$ and $m = |O(t)|$. For a transition t, $I(t)$ is called the input set of t and $O(t)$ is called the output set of t. A transition t is fired only when every place in its input set has at least one token which can be removed in the process of firing. If a transition t's input set fulfilled this condition under a marking μ of a Petri net, t is enabled and can be fired. Otherwise, t is not enabled and cannot be fired. The input set $I(t)$ and output set $O(t)$ of a transition t can be empty and the two sets may have a nonempty intersection; that is, a place may exist in the input set $I(t)$ and the output set $O(t)$ at the same time. If the input set $I(t)$ is empty, transition t can be fired under any markings. If the output set $O(t)$ is empty, transition t will not add tokens to any place.

For the need of graphical representation, the mapping from a place to a transition is defined as an arc. There is an arc from each place in the input set $I(t)$ of a transition t to t and an arc from each place in the output set $O(t)$. The set of all arcs in a net is a relation on $P \times T$ and $T \times P$, where, for every place p_i in the input set $I(t)$ of a transition t, $(p_i, t) \in A$ and, for every place p_o in the output set $O(t)$, $(p_o, t) \in A$.

A Petri net can be represented in a graphical form by a bipartite graph. In a Petri net graph, eclipses represent places, rectangles represent transitions, and connections between eclipses and rectangles represent arcs. An example of graphically represented Petri net is shown in Figure 1. Definitions of P, T, and A of this net are as follows:

$$\begin{gathered} P = p_1, p_2, p_3, \\ T = t_1, t_2, \\ A = (p_1, t_1), (p_2, t_1), (t_1, p_3), (p_3, t_2). \end{gathered} \tag{1}$$

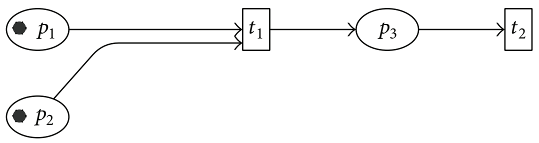


FIGURE 1: The net in initial state.

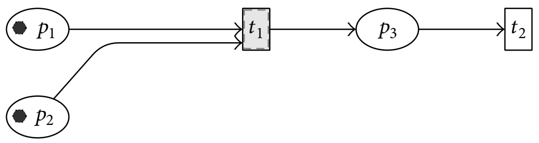


FIGURE 2: Transition t_1 enabled.

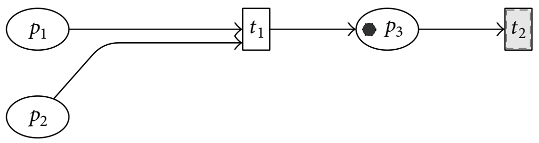


FIGURE 3: Transition t_2 enabled.

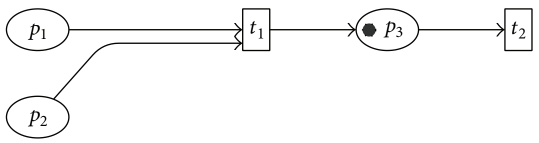


FIGURE 4: The model in deadlock.

In the initial state of the Petri net, p_1 and p_2 both contain a token, so the initial marking of the net is 1, 1, 0 and transition t_1 is enabled according to aforementioned rules, while transition t_2 is not enabled due to the fact that place p_3 in its input set is empty. This state is illustrated in Figure 2 where the enabled transition, that is, t_1, is highlighted.

If transition t_1 fires, it removes a token from place p_1 and a token from p_2 and adds a token to place p_3. After transition t_1's firing, the Petri net goes to a new marking as shown in Figure 3, which is 0, 0, 1. Now, transition t_2 is enabled and transition t_1 is no longer enabled. If transition t_2 fires, it removes a token from place p_3 and does not generate a token to any place. As shown in Figure 4, all of the Petri net's places are empty now and no transition in the net can fire any more. If no transition is enabled under a given marking of a Petri net, this net is in a deadlock state.

2.2. Coloured Petri Nets. To enhance the expressiveness of the pure Petri nets formalism, we have depicted above that many variants are designed including prioritized Petri nets (PPN) [16], timed Petri nets (TTN) [17], and probabilistic Petri nets (PPN) [18]. One of these extensions is coloured Petri nets (CPN) [14], whose main contribution is to extend the pure Petri nets' type system to a more elaborate one. In a pure Petri net, a token is simply a place holder which means that a place has a unit of data that can be absorbed by a transition, but the value of the token is ignored. A place is defined with a colour which is the type of the place and all tokens residents

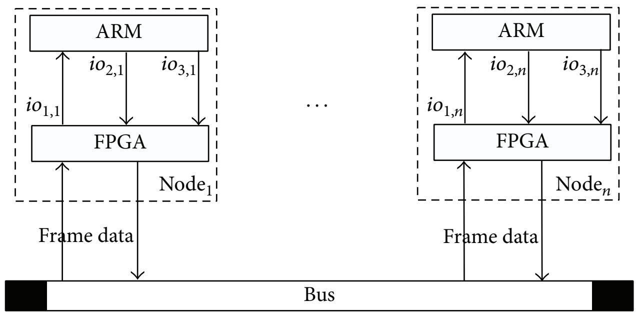


FIGURE 5: Representation of system.

in this place should be a value of the predefined type. A detailed explanation of its subtle execution semantics will be illustrated in the following demonstration and analysis of a set of CPN models.

3. Modeling of a Heterogeneous Embedded System

3.1. An Embedded System of Vehicle Protocol. Our system is an embedded system which implements the vehicle protocol *IEC-61375* [19]. The system consisted of a set of nodes; all connect to a bus which transfers frame data between these nodes, as shown in Figure 5. Each node consists of an ARM board and an FPGA board. An FPGA board may send frame data to the bus or receive frame data from the bus. It also communicates with an ARM board through GPIO ports in order to transfer messages between each other. An ARM board only interacts with its related FPGA board. There are three GPIO ports which are as follows.

(i) $io_{1,k}$ is used by the FPGA system to send a master frame interrupt to the ARM system in node k.

(ii) $io_{1,k}$ is used by the ARM system to notify the FPGA system of the arrival of a master message in node k.

(iii) $io_{1,k}$ is used by the ARM system to send a master message to the FPGA system in node k.

Among all nodes, there is a master node who monitors the whole network and all other left nodes are slave nodes under the master node's supervision. During a typical work flow, the master node polls slave nodes for new status data according to a predefined order which is stored in the ROM of the master node's ARM board part. This process is called a polling process and a success retrieval of a slave node's status by the master node is called a round of poll. Usually, a number is used to identify the node in order to decide the source and destination of each frame. The master node is numbered **1** under the typical configuration of such an embedded system and slave nodes are numbered from 2. In the system shown in Figure 5, Node_1 is the master node and Node_2 to Node_n are slave nodes.

To illustrate how nodes communicate with each other, we discuss the polling process of the master node as an example in Figure 6. In the system introduced above, the ARM system is responsible for sending poll messages. As a result, all slave nodes have their FPGA part participated in the polling process and the master node has both the ARM part and the FPGA part involving in the polling process. In the following discussion, only three nodes are involved for simplification among which Node_1 is the master node while Node_2 and Node_3 are slave nodes. We let A_i $(i = 1, 2, 3)$ denote the ARM board of each node and F_i $(i = 1, 2, 3)$ the FPGA board. The bus is denoted by $Bus : Line$. The messages sent between the ARM system and the FPGA system are labeled as follows.

(i) **MF_INTR** is the master frame interrupt from the FPGA system to the ARM system.

(ii) **MF_Flag** is the notification signal from the ARM system to the FPGA system.

(iii) **MF_Data** is the poll message from the ARM system to the FPGA system.

In this demo, Node_1 is the master node, so the polling process starts from A_1 and F_1. F_1 first triggers a master frame interruption of A_1 through $io_{1,1}$ of the ARM board. A_1 then handles this interruption and starts polling the slave nodes in the network in a predefined order. A_1 sets $io_{2,1}$ to high level to notify F_1 the coming of a polling message, and then A_1 puts the content of a polling message with value **2** which means the message's target is Node_2) to $io_{2,1}$. F_1 is triggered by the high level of $io_{2,1}$ and encapsulates the message received from A_1 to a frame which has the format of (**type**: **poll/response**, **source**, and **target**). This frame is put on the bus in a broadcast way, meaning that all nodes (including the sender of the frame) connected to the bus know the existence of this frame. But only the frame's target will handle the content of it, and all other nodes will neglect the frame's content automatically. So, F_2 receives this frame from the bus and sends out its response frame back to the bus which targets Node_1. When F_1 gets the response frame from F_2, it triggers A_1 again to start another polling which targets Node_3.

3.2. Details of the System. The GPIO interface between the ARM board and the FPGA board works as an intermediate data store for the interaction of a sequential program written in C and a parallel program written in VHDL. We demonstrate an abstract version for the C program in Algorithm 1 and an abstract version for the VHDL program in Algorithm 2. The C code in Algorithm 1 is an interrupt handler which sends a master frame numbered from **2** to **SLAVE_COUNT** in a roll. The VHDL code in Algorithm 2 checks **mf_flag** every cycle and transfers the value from **mf_data** to **target_node**. The following CPN model is based on these codes.

Figure 7 shows a CPN model of the ARM system which receives a master frame interrupt and sends a poll message to the FPGA system. Figure 8 shows a CPN model of the FPGA system which interrupts the ARM system and transfers the polling information to the bus. Colour sets used in both models are defined in Table 1 which contains a list of places of the specified colour set. All related variables are listed in Table 2.

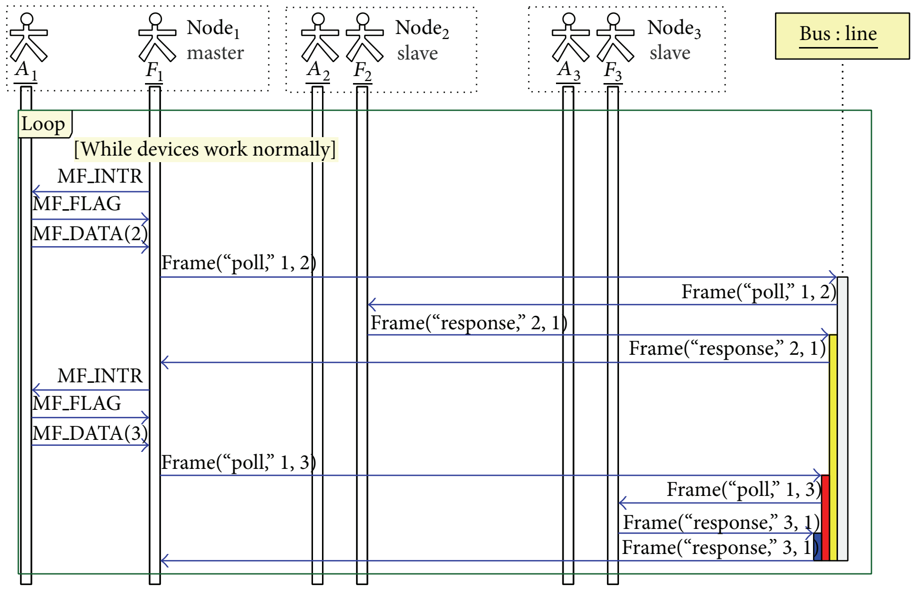


Figure 6: Message chart of polling process.

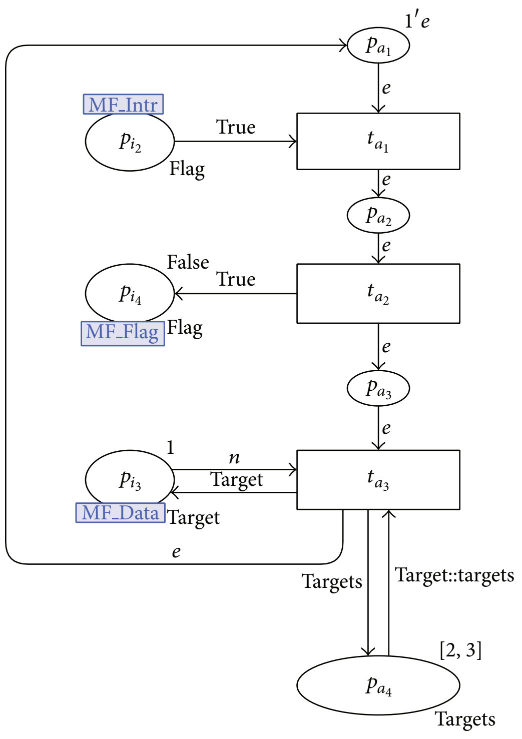


Figure 7: The ARM system's CPN model.

3.2.1. The ARM System. In the CPN model of the ARM system, the meanings of places are as follows.

(i) Places that represent states of the ARM system are p_{a_1}, p_{a_2}, and p_{a_3}, all of which have type *State*. Because tokens in these places can only have value **e**, we call these tokens *unit tokens*.

(ii) p_{i_2} and p_{i_4} are places with type *Flag*. p_{i_2} represents the master frame interrupt and p_{i_4} the signal for the coming master frame data.

(iii) Place p_{i_3} with type *Target* is used to store the master frame data sent from the ARM system to the FPGA system.

(iv) Place p_{a_4} with type *Targets* stores a list of numbers which represents the order of slave nodes to be polled.

In the initial state of the system, a unit token is located in p_{a_1}. t_{a_1} is enabled for the unit token in p_{a_1} and the token with value **true** in p_{i_4}. When t_{a_1} fires, it removes a unit token from p_{a_1} and a token from p_{i_4} and generates a unit token to

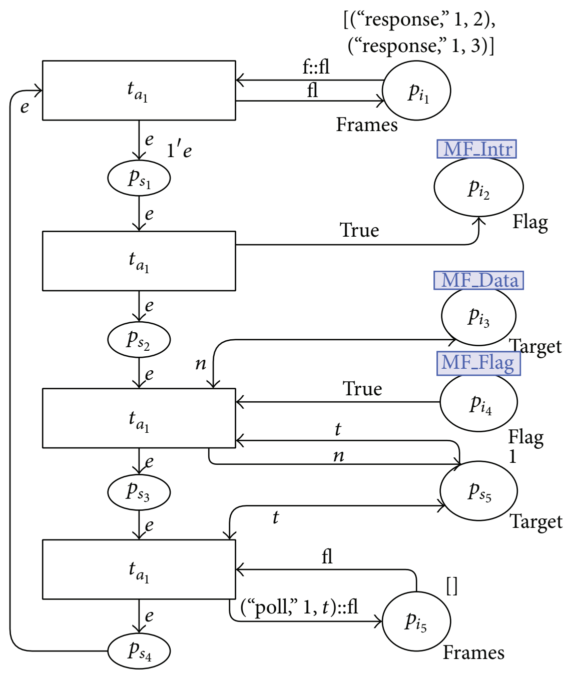


Figure 8: The ARM system's CPN model.

```
void master_frame_interrupt_handle ()
{
    node_number++;
    if (slave_node_number >= SLAVE_COUNT)
    {
        slave_node_number = 2;
    }
    arm_set_main_frame_flag (TRUE);
    arm_send_main_frame_data (node_number);
}
```

Algorithm 1: C code on the ARM system.

```
process (reset, clk, current_state,
        mf_flag, mf_data)
    begin
        if reset = '1' then
            current_state <= start;
        elsif clk'event and clk = '1' then
            if mf_flag = '1' then
                target_node <= mf_data;
            current_state <= next_state;
            end if;
        end if;
end process;
```

Algorithm 2: VHDL code on the FPGA system.

place p_{a_2}. Then, t_{a_2} is enabled immediately when p_{a_2} receives a unit token. t_{a_2}'s firing removes a unit token from place p_{a_2} and then puts a unit token to p_{a_3} and a token with value **true** to p_{i_4}. The token with value **true** in p_{i_4} indicates the FPGA system of a master frame poll message from the ARM system. Then, t_{a_3} is enabled due to the unit token in p_{a_3} and the token in p_{i_3}. t_{a_3} has three input places: p_{a_3}, p_{a_4}, and p_{i_3}. The firing of t_{a_3} consumes a unit token from p_{a_3}, a token with type *Target* from p_{i_3}, and a token with type *Targets* from p_{a_4}. The *Targets* token from p_{a_4} is bound to two variables: **target** which corresponds to the number of the slave node being polled currently and **targets** which corresponded to numbers of remaining slave nodes waiting to be polled. The firing of transition t_{a_3} sends a unit token to p_{a_1} indicating the change of system state, a *Target* token called **target** to p_{i_3} indicating the slave node's number to be polled and a *Targets* token to p_{a_4} with numbers of remaining slave nodes to be polled in the coming polls. When t_{a_3} finishes its firing, the ARM system waits the master frame interrupt again to start a new poll.

3.2.2. The FPGA System. In the CPN model of the FPGA system, the meanings of places are as follows.

(i) Places that represent states of the FPGA system are p_{s_1}, p_{s_2}, p_{s_3}, and p_{s_4} all of which have type *State*. Tokens in these places are *unit tokens* just like those in the ARM system's state places.

(ii) p_{i_2}, p_{i_3}, and p_{i_4} are the same places in Figure 7.

(iii) p_{i_1} and p_{i_5} simulate the frames sent and received between the bus and the FPGA board. p_{i_1} represents frames from the bus and p_{i_5} represents frames to the bus.

Table 1: Colour set definitions.

Colour set	Definition	Places
State	*Unit* **withe**	$p_{a_1}, p_{a_2}, p_{a_3}, p_{s_1}, p_{s_2}, p_{s_3}, p_{s_4}$
Flag	*BOOL*	p_{i_2}, p_{i_4}
Frame	**product** *STRING* × *INT* × *INT*	
Target	*INT*	p_{s_5}, p_{i_3}
Frames	**list** *Frame*	p_{i_1}, p_{i_5}
Targets	**list** *Target*	p_{a_4}

Table 2: Variables in model.

Variables	Colour set
n, **t**	*INT*
flag	*BOOL*
f	*Frame*
fl	*Frames*
target	*INT*
targets	*Targets*

(iv) Place p_{s_5} with type *Target* stores the number of slave node that will be used to construct a poll frame.

In the initial configuration, p_{s_1} contains a unit token, so t_{s_1} outputs a token with value **true** to p_{i_2} and a unit token to p_{s_2}. The system of ARM treats a token with value **true** in p_{i_2} as a trigger to start its master frame process. t_{s_2} has four input places: p_{s_2}, p_{i_3}, p_{i_4}, and p_{s_5}. p_{i_3} and p_{s_5} already contain a token in each of them, so when the ARM system gives p_{i_4} a token with value **true**, t_{s_2} is enabled. p_{i_3} offers t_{s_2} a *Target* token which is bound to variable **n** indicating the number of slave node to be polled next. p_{s_5} is updated in t_{s_2}'s firing and a *Target* token with value **n** replaces the old value in this place. p_{s_3} gets a unit token, meaning the change of the FPGA system's state. t_{s_3} retrieves a *Target* token **t** from p_{s_5}, composes a poll frame in the form of (**poll**, **1**, **t**), then appends this frame to the *Frames* token from p_{i_5}, and sends an updated token back to this place. A unit token is sent to p_{s_4} by t_{s_3} and the FPGA system goes to the monitoring state. When p_{i_3} has tokens, t_{s_4} fires to remove the head of the **Frames** token in p_{i_1} and sends a unit token to p_{s_1}. The FPGA system goes back to its initial state and starts another master frame poll process of the ARM system.

4. Analysis of the Model

4.1. Review of the Model. The FPGA system cyclically checks the state of master frame flag on the port between the ARM system and itself and once the value of the flag becomes **true** in a cycle, the FPGA system tries to retrieve new master frame data from the corresponding port immediately. As shown in Algorithm 2, the FPGA system gets master frame data and sends the data to a signal called **target_node** that represents the current number of the slave node to be polled. Then, the value of **target_node** is used to construct a poll frame sent to the bus.

According to the designer's intention, the value in **target_node** should be updated every time the FPGA system retrieves master frame data. But t_{s_2} in Figure 8 updates the token in t_{s_5} according to the token in t_{i_2}. However, the process of updating may be ineffective and, as a result, the token in t_{s_5} contains a stale value. To explore this scenario in detail, we focus on the interaction between the ARM system and the FPGA system. There are five transitions which participate in the interaction:

(i) t_{s_1} and t_{s_2} of the FPGA system,

(ii) t_{a_1}, t_{a_2}, and t_{a_3} of the ARM system.

t_{s_1}, t_{a_1}, and t_{a_2} always execute in an order according to the strict dependency caused by tokens' generation and consumption in p_{i_2} and p_{i_4}. So, the execution trace of the three transitions is decidable. The other two transitions which are involved in the interaction act in a different manner. After t_{a_2} fires, a token with value **true** appears in p_{i_2} and a unit token appears in p_{a_3}, which activates t_{s_2} and t_{a_3}, respectively. Normally, t_{s_2} should fire after t_{a_3}, so that the FPGA system can construct a right frame which is sent to a slave node. However, the nature of C programs running on an ARM platform is asynchronous sequential execution, while the nature of VHDL programs running on an FPGA platform is synchronous concurrent execution. Usually, the FPGA system will complete the check of the master frame flag and retrieval of the master frame data in the same cycle. So, possibility exists that the FPGA system retrieves the value of the master frame data before its value is updated by the ARM system and thus gets a token with a stale value. We cannot predicate when will the ARM system updates the value of the master frame data after the value of the master frame flag is set to **true** for the undecidability of the operating system's scheduling scheme. So, if the FPGA system's new execution cycle starts before the value of master frame data is set by the ARM system, a timing mismatch happens. To verify the analysis of the data inconsistency in the CPN model, we apply the approach of model checking on it in the following section.

4.2. Property Analysis. To analyze the behavior of a modeled system, a state-space based approach can be used to dive into the details of models and get a thoroughly understanding of the system. A state of a CPN model is the marking of the model, that is, the number and value of tokens in each place in the whole model, and the initial marking of a model is its initial state. A CPN model changes its state when a transition of the model fires. All the states that a CPN model possibly accesses through a series of transition firing are called the state space of the model which may be finite or infinite. A property formula is an assertion about the state space of a model that whether the property holds in all states of the model and is usually called *property* for short. Model checking is an approach that checks whether a property about a model is fulfilled. If the property is not fulfilled, the model checking algorithm gives a counter example which violates the property. There are many description formalisms for properties, such as *CTL*, *LTL*, and *CTL** (add some refers) and many model checking algorithms target these

Table 3: Property related functions.

Function	Type
isStale	$Transition \rightarrow BOOL$
myASKCTLformula	$Node \rightarrow BOOL$
eval_node	$(Node \rightarrow BOOL) \rightarrow BOOL$

formalisms. In the field of CPN, *ASK-CTL* is used to describe the property of a CPN model. CPN tools implement the representation of *ASK-CTL* formulas and its model checking algorithms in SML language. A state is also called a node because the state space is represented as a state graph in which an edge represents a firing of a transition and a node represents a marking of the model. In the following statements, we will refer to states as nodes when needed.

Let A be the abstract *ASK-CTL* formula type; then the following formulas with type $A \rightarrow A$ are considered.

(i) *POS* is a node formula which is true if it is possible, from the current node, to reach a node where A is true.

(ii) *INV* is a node formula which is true if A is true for all reachable nodes from the current node.

(iii) *EV* is a node formula which is true if A eventually becomes true from the current node.

(iv) *ALONG* is a node formula which is true if there is a path for which A is true for every node.

The CPN model of the master node system has the potential to enter a node in which there is a data mismatch caused by interface incompatibility. This kind of defect can be expressed as:

"when transition t_{a_2} fires, t and n which bind to the transition are equal in value."

So, if this property is true in a node, that is, the value of master frame data read by the FPGA system is equal to the value of **target_node** got in previous cycles, a data mismatch appears. This property can be coded as *ASK-CTL* formula in SML form.

The types of these variables/functions are listed in Table 3.

isStale is a transition function which returns **true** when the parameter is a transition equal to t_{a_2} bound with $n = t$ where $n = 2, 3$ in this case. myASKCTL-formula is a node function which takes a node as the only parameter and returns **true** when there is a node in which isStale is evaluated to be **true**. eval_node is a function which takes a node function as the property formula and a node as the starting node and returns **true** when the application of the formula on the node returns **true**. In this case, we evaluate the property written in SML code above and get the following result: **var it** = **true**: *BOOL*, which means the property holds with respect to the initial state in state space (see Algorithm 3); that is, there is a node in which p_{s_5} gets a token with a stale value.

```
fun isStale a =
    (Bind.FPGA'Receive_MF_Data
        (1, {n = 3, t = 3})
    = ArcToBE a) orelse
    (Bind.FPGA'Receive_MF_Data
        (1, {n = 2, t = 2})
    = ArcToBE a);
val myASKCTLformula =
    POS(MODAL(AF("stale value", isStale)));
eval_node myASKCTLformula InitNode;
```

Algorithm 3: Propery formula in SML code.

5. Conclusion

We will not blame a C programmer or a VHDL programmer for writing the code shown in Algorithms 1 and 2, since they both obey the common paradigm in their respective domain. However, in development of a heterogeneous system, a programmer should be more alert and pay special attention to the situation when a normal assumption is not fulfilled by the guarantee of other domains in which a defect in the resulted system may occur. Many approaches can be used to reduce this potential risk and one of them is model checking method. In this paper, we demonstrate a heterogeneous system which is composed of two subsystems: an ARM system and an FPGA system. One of the subsystems behaves synchronously and the other asynchronously. Through a state based analysis way, we find defect in the design of the system which may lead to data inconsistency between the two subsystems. Then, our analysis is verified by constructing a property formula which is computed by a model checking program.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

References

[1] N. Halbwachs, P. Caspi, P. Raymond, and D. Pilaud, "The synchronous data flow programming language LUSTRE," *Proceedings of the IEEE*, vol. 79, no. 9, pp. 1305–1320, 1991.

[2] D. Gay, P. Levis, R. von Behren, M. Welsh, E. Brewer, and D. Culler, "The nesC language: a holistic approach to networked embedded systems," *ACM Sigplan Notices*, vol. 38, no. 5, pp. 1–11, 2003.

[3] E. A. Lee, "Cyber physical systems: design challenges," in *Proceedings of the 11th IEEE Symposium on Object/Component/Service-Oriented Real-Time Distributed Computing (ISORC '08)*, pp. 363–369, May 2008.

[4] S. Vercauteren, B. Lin, and H. de Man, "Constructing application-specific heterogeneous embedded architectures from custom HW/SW applications," in *Proceedings of the 1996 33rd Annual Design Automation Conference*, pp. 521–526, IEEE, June 1996.

[5] S. Kumar, *The Codesign of Embedded Systems: A Unified Hardware/Software Representation*, Springer, 1996.

[6] B. Schatz, A. Pretschner, F. Huber, and J. Philipps, “Model-based development of embedded systems,” in *Advances in Object-Oriented Information Systems*, pp. 298–311, Springer, 2002.

[7] M. Törngren, D. Chen, and I. Crnkovic, “Component-based vs. model-based development: a comparison in the context of vehicular embedded systems,” in *Proceedings of the 31st EUROMICRO Conference on Software Engineering and Advanced Applications (EUROMICRO-SEAA ’05)*, pp. 432–440, IEEE, September 2005.

[8] D. C. Schmidt, “Guest editor’s introduction: model-driven engineering,” *Computer*, vol. 39, no. 2, pp. 25–31, 2006.

[9] S. Edwards, L. Lavagno, E. A. Lee, and A. Sangiovanni-Vincentelli, “Design of embedded systems: formal models, validation, and synthesis,” *Proceedings of the IEEE*, vol. 85, no. 3, pp. 366–389, 1997.

[10] P. Marwedel and G. Goossens, *Code Generation for Embedded Processors*, vol. 11, Kluwer Academic Publishers, 1995.

[11] A. Fin, F. Fummi, M. Martignano, and M. Signoretto, “SystemC: a homogenous environment to test embedded systems,” in *Proceedings of the 9th International Symposium on Hardware/Software Codesign*, pp. 17–22, ACM, April 2001.

[12] Y. Jiang, H. Zhang, X. Song et al., “Bayesiannetwork-based reliability analysis of plc systems,” *IEEE Transactions on Industrial Electronics*, vol. 60, no. 11, pp. 5325–5336, 2013.

[13] “BPDF: a statically analyzable dataflow model with integer and boolean parameters,” in *Proceedings of the 11th ACM International Conference on Embedded Software (EMSOFT ’13)*, V. Bebelis, P. Fradet, A. Girault, and B. Lavigueur, Eds., pp. 1–10, September 2013.

[14] K. Jensen, “Coloured petri nets,” in *Petri Nets: Central Models and Their Properties*, pp. 248–299, Springer, 1987.

[15] H.-M. Hanisch, J. Thieme, A. Lueder, and O. Wienhold, “Modeling of PLC behaviour by means of timed net condition/event systems,” in *Proceedings of the IEEE 6th International Conference on Emerging Technologies and Factory Automation (ETFA ’97)*, pp. 361–369, September 1997.

[16] G. Balbo, “Introduction to stochastic petri nets,” in *Lectures on Formal Methods and Performance Analysis*, pp. 84–155, Springer, 2001.

[17] L. Popova-Zeugmann, “Timed petri nets,” in *Time and Petri Nets*, pp. 139–172, Springer, 2013.

[18] M. A. Marsan, “Stochastic petri nets: an elementary introduction,” in *Advances in Petri Nets 1989*, pp. 1–29, Springer, 1990.

[19] C. Schaefers and G. Hans, “Iec 61375-1 and uic 556-international standards for train communication,” in *Proceedings of the IEEE 51st Vehicular Technology Conference (VTC ’00)*, vol. 2, pp. 1581–1585, IEEE, Tokyo, Japan, May 2000.

Hindawi Publishing Corporation
Journal of Applied Mathematics
Volume 2014, Article ID 969523, 7 pages
http://dx.doi.org/10.1155/2014/969523

Research Article

An Efficient Multitask Scheduling Model for Wireless Sensor Networks

Hongsheng Yin,[1] Honggang Qi,[2] Jingwen Xu,[2,3] Xin Huang,[1] and Anping He[4]

[1] *China University of Mining & Technology, Xuzhou 221116, China*
[2] *University of Chinese Academy of Sciences, Beijing 101408, China*
[3] *Institute of Electrical Engineering, Chinese Academy of Sciences, Beijing 100190, China*
[4] *Guangxi Key Laboratory of Hybrid Computation and IC Design Analysis, Guangxi University for Nationalities, Nanning 530006, China*

Correspondence should be addressed to Honggang Qi; hgqi@ucas.ac.cn and Anping He; hapetis@gmail.com

Received 5 January 2014; Accepted 19 February 2014; Published 30 March 2014

Academic Editor: X. Song

The sensor nodes of multitask wireless network are constrained in performance-driven computation. Theoretical studies on the data processing model of wireless sensor nodes suggest satisfying the requirements of high qualities of service (QoS) of multiple application networks, thus improving the efficiency of network. In this paper, we present the priority based data processing model for multitask sensor nodes in the architecture of multitask wireless sensor network. The proposed model is deduced with the M/M/1 queuing model based on the queuing theory where the average delay of data packets passing by sensor nodes is estimated. The model is validated with the real data from the Huoerxinhe Coal Mine. By applying the proposed priority based data processing model in the multitask wireless sensor network, the average delay of data packets in a sensor nodes is reduced nearly to 50%. The simulation results show that the proposed model can improve the throughput of network efficiently.

1. Introduction

Wireless sensor network (WSN) is a basic network for accessing the data information in the sensor layer of the Internet of Things (IOS). WSN is widely applied in various areas [1]. For instance, in military, the troop and equipment can be identified and services can be coordinated to fight with the assistance of WSN. In the aspect of biomedical, human health can be monitored by the surgical sensors implanted in body, which is a typical application of WSN. Moreover, in earthquake prediction, ad hoc deployment of seismic sensors along the volcanic area can detect the development of earthquakes and eruptions [2]. WSN integrates the technologies of information sensing, data processing and transmission, which is a multitask system. Numerous data services are operating on the multitask system, such as the wireless monitoring and information management systems for coalmine safety production. The types of the service data provided by WSN are classified as automatic control command, safety monitoring data, audio and video data, and so on [3]. Usually, the coverage range of wireless sensor network is not very large. Thus, the transmission delay of electromagnetic wave may be neglected. As the sensor nodes are constrained in computation, storage, and energy, it is difficult to meet the requirement of good quality of service (QoS) for more tasks running in a network. Moreover, due to the unreliable wireless channel interfered by noise, QoS of the wireless transmission is often depressed, which is especially significant in multitask wireless network. And therefore, in order to improve the performance of multitask wireless sensor network, it is very important to carry out research on the high-efficient multitask scheduling model for wireless sensor network.

TinyOS is an operating system, which is widely used in wireless sensor networks. The operating system adopts First Come First Served (FCFS) scheduling strategy for task scheduling, which is efficient to reduce the requirements of storing space [4, 5]. However, as there are no the priorities among various kinds of service data, some real-time services cannot be timely responded, so that many services are missed,

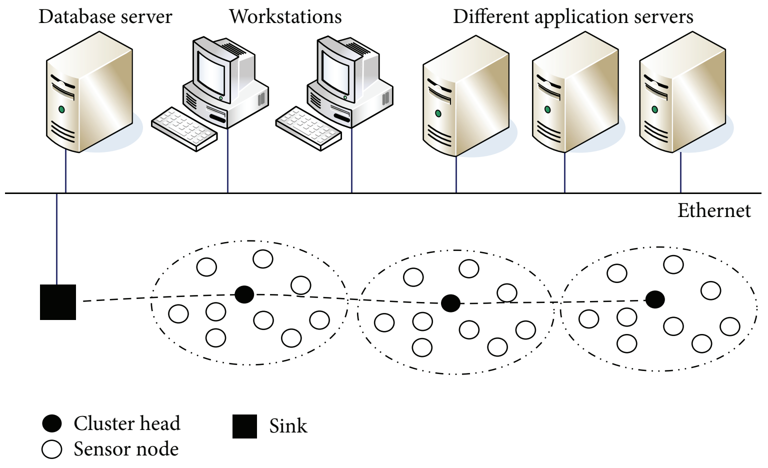


Figure 1: Multiapplication system architecture of wireless sensor network.

which results in the low throughput of network [6]. For the drawback of TinyOS in the scheduling strategy, the researchers have done many researches for improving the scheduling strategy. The contribution [7] introduced a dual circular-based task scheduling strategy. In this strategy, the single circular queue is substituted with the dual circular queues with different priorities. The tasks are assigned different priorities and then are allocated in the two circular queues according to their priorities. The tasks in different queues are dynamically switched according to their time variations for guaranteeing them to be responded as much as possible. The strategy improves the speed of response to real-time tasks, but the throughput of network is still low. In contribution [8, 9], a priority based soft real-time task scheduling strategy was proposed, which increases the throughput of network but does not satisfy the real-time requirement of some high-priority tasks. For solving the existing problem in [8, 9], the contribution [10] introduced an improving scheduling strategy, EF-RM (emergency task first rate monotonic), which is the preemptive scheduling for both periodical and nonperiodical tasks to ensure the implementation of the important task of priority in TinyOS. The contribution [11] proposed the IS-EDF (idle sleep-earliest deadline first) scheduling strategy, which adjusts the priority of tasks dynamically to ensure that the important task is real-time processed.

In this paper, through further research on the relevant contributions mentioned above, we propose the priority queue-based data processing model for multitask network and deduce the theoretical formulas of the QoS of network with the proposed model, including average queue length, delay, and delay jitter. The performance of the proposed models is analyzed and compared by the practical simulation experiments.

The rest of this paper is organized as follows: The architecture of multitask wireless sensor network is presented in Section 2. Then, the queue theory is introduced in Section 3 first. Subsequently, in Sections 4 and 5, two queue models are described, respectively. The experimental results are shown in Section 6. Finally, we conclude this paper in Section 7.

2. Architecture of Multitask Wireless Sensor Network

In the wireless sensor network, a large number of wireless sensor nodes are densely and fully deployed in the network. These wireless sensor nodes are organized into many clusters. Each cluster is composed of a cluster head and multiple sensor nodes. The internal sensor nodes can communicate with each other in the cluster. The external communications between clusters are fulfilled by the cluster heads in these clusters. Moreover, the cluster head is responsible for assigning the time slot for each sensor nodes in its cluster. The data collected from each wireless sensor nodes are first gathered in the cluster heads and then transmitted to a database in the server by the sink nodes through the wired Ethernet. All applications in the network share the data in the database for different functions. The architecture of multitask wireless sensor network is shown in Figure 1.

3. Concept of Queuing Theory

Queuing theory is a mathematical method for analyzing the congestion and delays of data packets in a link. With queuing theory, the arrival, service, and depart of data packets can be accurately evaluated so that the data packets can be efficiently scheduled in a link. For describing the proposed model based on the queuing theory easily, we give the following definitions.

Definition 1 (inputting distribution $A(t)$). In the inputting process, let C_n be the nth data packet arriving at the network node and the arrival time is τ_n; then, $t_n = \tau_n - \tau_{n-1}$, which means the time interval between C_n and C_{n-1}. Assume that $\tau_0 = 0$ and the arriving data packets are independent; then,

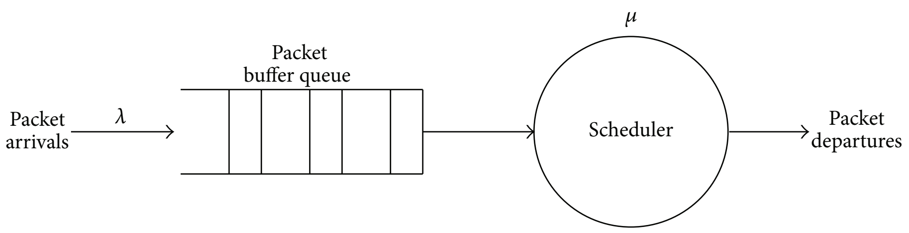


Figure 2: Data processing model for the queuing system without priority.

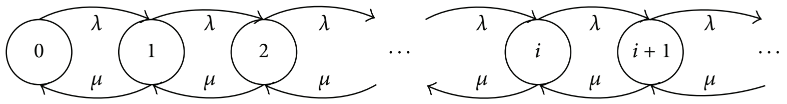


Figure 3: State diagram of birth-death process for queuing system without priority.

$\{t_n\}$ is the sequence of independent random variables, written as $A(t)$.

Definition 2 (serving distribution $B(t)$). In the service process, let the service time of data packet C_n be ν_n. Assume that the services of data packets are independent; then, $\{\nu_n\}$ is the dependent sequence of random variables, written as $B(t)$.

Definition 3 (arrival probability of data packet $p(n)$). Let $N(t)$ be the number of data packets in a network node at time t and let $p(n)$ be the arrival probability of n data packets in time interval (t_1, t_2); then, there is the relation

$$p(n) = P\{N(t_2) - N(t_1) = n\}, \quad (t_2 > t_1, n \geq 0). \tag{1}$$

Definition 4 (arrival rate of data packet λ). An average number of data packets arrive at a network node in unit time, which reflects how fast the data packets arrive at a network node. $1/\lambda$ is just the average arrival time interval of data packets.

Definition 5 (service rate of network node μ). An average number of served data packets depart from a network node in unit time, which reflects how fast the services are in the network node. $1/\mu$ is just the average time of the data packets severed in a network node.

Definition 6 (service intensity ρ). The average service time of each network node in unit time, which is an important indicator for measuring how busy the network nodes, is $\rho = \lambda/\mu$, $0 \leq \rho < 1$.

In the real situation of wireless sensor network, the data packets arrive at sensor nodes continuously. Thus, the number of data packets is regarded as infinite. For simplicity, the arrival times of data packets are assumed to follow M/M/1 queue model. The input process of data packets, that is, the arrival times, is similar to the Poisson stream with parameter λ. The arriving time interval $A(t)$ and service time $B(t)$ follow the negative exponential distribution with parameters λ and μ, respectively, where the service window size is 1. Based on the reasonable assumptions and the definitions on queue theory mentioned above, two queue system models, nonpriority and priority models, are analyzed and compared as follows. And therefore, the high efficient queue model is proposed in this paper.

4. Data Processing Model Based on Nonpriority Queue System

As shown in Figure 2, the data packets enter the network nodes continuously and are lined up in a queue with the average arrival rate λ. The data packets depart in turn from the queue and data services are scheduled in the scheduler at the average processing rate μ. The node state N at time t is denoted as $N(t) = (i)$, where i is the number of data packets including the processing data packet, that is, the queue length. It is easy to be proved that $\{N(t), t \geq 0\}$ is birth-death process [12–16].

Let $p(i;t) = P\{N(t) = (i)\}$, where $p(i) = \lim_{t\to\infty} p(i;t)$, $i \geq 0$. Referring to Figure 3, if $\rho = \lambda/\mu < 1$, the balance equations are as follows:

$$\begin{gathered}\lambda p(0) = \mu p(1) \\ (\lambda+\mu) p(1) = \lambda p(0) + \mu p(2) \\ (\lambda+\mu) p(2) = \lambda p(1) + \mu p(3) \\ \cdots \\ (\lambda+\mu) p(i) = \lambda p(i-1) + \mu p(i+1).\end{gathered} \tag{2}$$

Because there is $\sum_{i=0}^{\infty} p(i) = 1$ and $p(i) = (1-\rho)\rho^i$ holds, so the average length of data packets in network node is.

$$Q = \sum_{i=0}^{\infty} ip(i) = \sum_{i=0}^{\infty} i(1-\rho)\rho^i = \frac{\rho}{1-\rho} \tag{3}$$

And the average waiting queue length of data packets in network node is

$$W = \sum_{i=0}^{\infty} ip(i+1) = \sum_{i=0}^{\infty} i(1-\rho)\rho^{i+1} = \frac{\rho^2}{1-\rho} \tag{4}$$

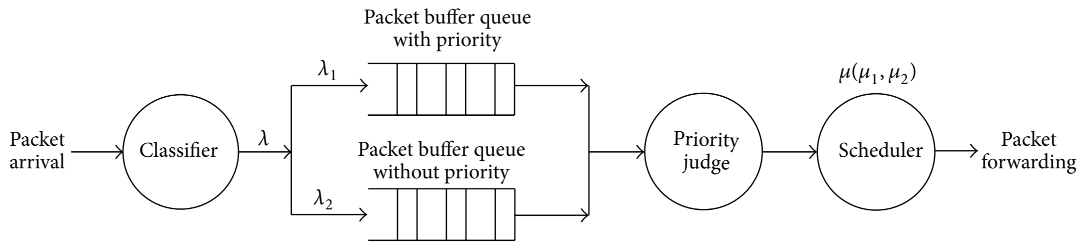


Figure 4: Data processing model for the queuing system with priority.

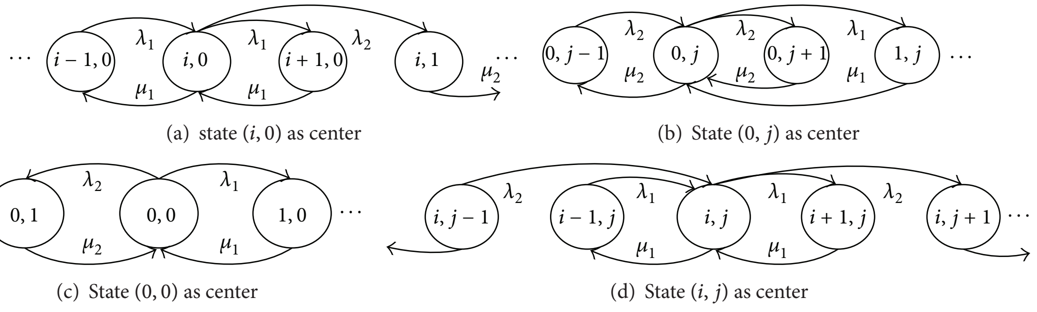


Figure 5: State diagram of birth-death process for queuing system with priority.

According to the Little theorem, the average waiting time of a data packet is expressed as

$$T_w = \frac{W}{\lambda} = \frac{\rho^2}{\lambda(1-\rho)} = \frac{\rho}{\mu(1-\rho)}. \tag{5}$$

And the average residence time of a data packet in network node, that is, delay of a data packet, is

$$T_Q = T_W + \frac{1}{\mu} = \frac{1}{\mu(1-\rho)} = \frac{1}{\mu-\lambda}. \tag{6}$$

And the delay jitter of a data packet in network node, that is, variance of delay, is

$$J_Q = \frac{1}{(\mu-\lambda)^2}. \tag{7}$$

5. Data Processing Model of Queue with Priority

In this model, the data packets entering the network node are classified into two queues with different priorities at the average rates λ_1 and λ_2 by the classifier, as shown in Figure 4. In the scheduler, according to the service rule given by the priority decision module, the services are obtained at the average processing rates μ_1 and μ_2. The priority decision module decides the processing sequence of data packets for the scheduler. It employs the preemptive priority service rule, which allows that the services of low-priority data packets are interrupted and free up resource for serving the high-priority data packets. The data packets with the same priority will be serviced according to the FCFS rule.

The data packet with priority is denoted by C1 and the data packet without priority is denoted by C2. The data packets C1 and C2 arrive at the network node in independent Poisson distribution with the parameters λ_1 and λ_2, respectively, and their service times follow the negative exponential distribution with the parameters μ_1 and μ_2. The system utilization is denoted by ρ, which is the time rate of service busy. That is the proportion of time that the scheduler busies. λ is the average arrival rate of all data packets and μ is the average processing rate for all data packets. The relations between these parameters can be expressed as $\lambda = \lambda_1 + \lambda_2$, $\rho = \rho_1 + \rho_2$, $\rho = \lambda/\mu$, $\rho_1 = \lambda_1/\mu_1$, and $\rho_2 = \lambda_2/\mu_2$.

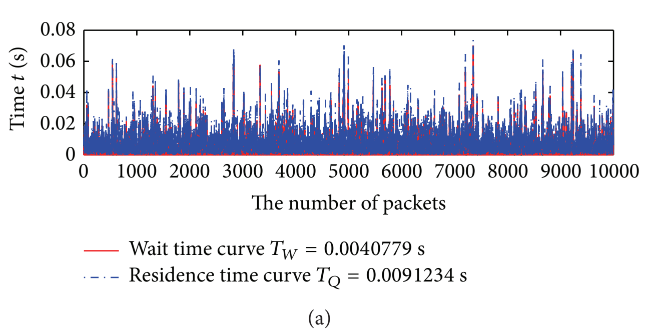
The state of network node at time t is denoted as $N(t) = (i, j)$. If the number of data packets C1 is i and the number of data packets C2 is j, it is easy to prove that $\{N(t), t \geq 0\}$ is the birth-death process [12–16]. The state diagram of birth-death process for queuing system with priority is shown in Figure 5.

Let

$$\begin{aligned} p(i,j;t) &= P\{N(t) = (i,j)\}, \\ p(i,j) &= \lim_{t\to\infty} p(i,j;t) \quad i,j \geq 0 \end{aligned} \tag{8}$$

According to the states in Figure 5, if $\rho = \rho_1 + \rho_2 = \lambda_1/\mu_1 + \lambda_2/\mu_2 \leq 1$, then the following equations hold:

$$\begin{aligned} (\lambda_1+\lambda_2)\, p(0,0) &= \mu_1 p(1,0) + \mu_2 p(0,1) \\ (\lambda_1+\lambda_2+\mu_1)\, p(i,0) &= \mu_1 p(i+1,0) \\ &\quad + \lambda_1 p(i-1,0) \quad i > 0, \\ (\lambda_1+\lambda_2+\mu_2)\, p(0,j) &= \lambda_2 p(0,j-1) + \mu_1 p(1,j) \\ &\quad + \mu_2 p(0,j+1) \quad j > 0, \end{aligned}$$



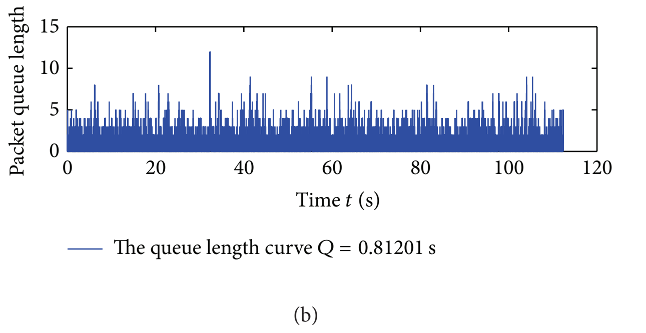


FIGURE 6: The curve of the time and queue length in the queuing model without priority.

$$(\lambda_1+\lambda_2+\mu_1)\,p(i,j)=\lambda_1 p(i-1,j)+\lambda_2 p(i,j-1)+\mu_1 p(i+1,j)\quad i,j>0. \tag{9}$$

The process of solving the equations (9) can be referred to [12–16], which solves $p(i,j)$ through the inverse solving method with the following generating function $\psi(u,z)$:

$$\psi(u,z)=\frac{(1-\rho_1-\rho_2)(1-z)\,\omega(z)}{[\rho_1 u\omega(z)-1]\{(\mu_1/\mu_2)[1-\omega(z)]z-(1-z)\,\omega(z)\}}, \tag{10}$$

where $\omega(z)=(\lambda_1+\mu_1+\lambda_2(1-z)-\sqrt{[\lambda_1+\mu_1+\lambda_2(1-z)]^2-4\lambda_1\mu_1})/2\lambda_1$. The solution of function $p(i,j)$ is solved by the differential generating function $\psi(u,z)$; that is,

$$p(i,j)=\frac{1}{i!j!}\cdot\left.\frac{\partial^{i+j}\psi(u,z)}{\partial u^i\partial z^j}\right|_{u=z=0}. \tag{11}$$

Let the probabilities of i C1 data packets and j C2 data packets in network node be $p_{i\bullet}$ and $p_{\bullet j}$, respectively. Their probabilities of generating functions are $\psi(u,1)$ and $\psi(1,z)$.

By formula (10), let $z\to 1$, using the L'Hospital Rule; we can get

$$\psi(u,1)=\frac{1-\rho_1}{1-\rho_1 u}=\sum_{i=0}^{\infty}(1-\rho_1)\,\rho_1^i u^i. \tag{12}$$

Thus, $p_{i\bullet}=(1-\rho_1)\rho_1^i$, which is the same as the M/M/1 queue system with only one kind of client. As a result, it shows that the existence of C2 data packets has no effect on the C1 data packets, which is in accord with the practical situation of network. Similarly, the average length of C1 data packet queue and the average length of C1 data packet waiting queue can be got as

$$Q_1=\frac{\rho_1}{1-\rho_1},\qquad W_1=\frac{\rho_1^2}{1-\rho_1}. \tag{13}$$

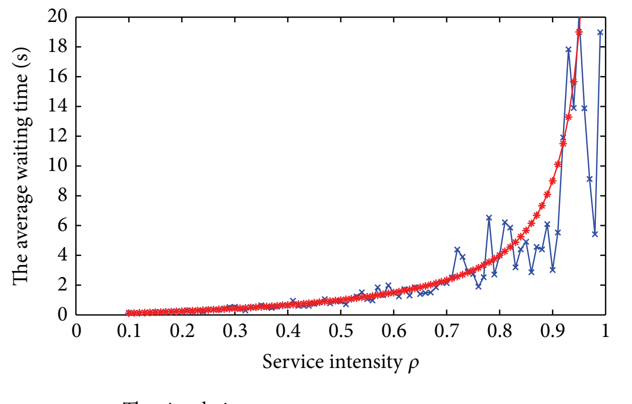


FIGURE 7: Theoretical and simulation curves of the average waiting time in the queuing model without priority.

And the average waiting time and average residence time of single C1 data packet are

$$T_{W_1}=\frac{\rho_1}{\mu_1(1-\rho_1)} \tag{14}$$

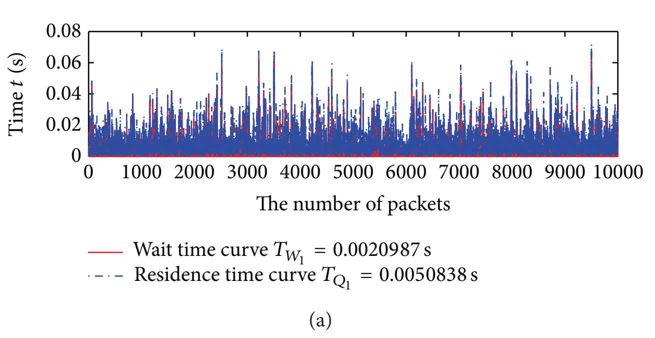
$$T_{Q_1}=\frac{1}{\mu_1(1-\rho_1)}=\frac{1}{\mu_1-\lambda_1}. \tag{15}$$

The delay jitter of a C1 data packet in the network node that is the delay variance is as follows:

$$J_{Q_1}=\frac{1}{(\mu_1-\lambda_1)^2}. \tag{16}$$

Then, by formula (10), we get

$$\psi(1,z)=\frac{(1-\rho_1-\rho_2)(1-z)\,\omega(z)}{[\rho_1\omega(z)-1]\{(\mu_1/\mu_2)[1-\omega(z)]z-(1-z)\,\omega(z)\}}. \tag{17}$$



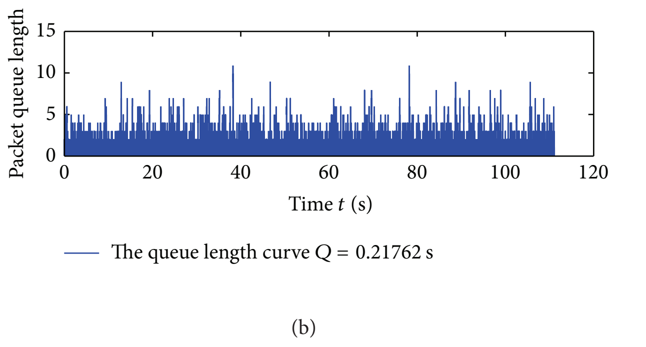


FIGURE 8: The curves of the time and queue length in the queuing model with priority.

Derivate formula (17) by z and then let $z = 1$; the average queue length of C2 data packets can be deduced as

$$Q_2 = \sum_{j=0}^{\infty} j p_{\bullet j} = \frac{\rho_2}{1-\rho_1-\rho_2}\left[1+\frac{\mu_2\rho_1}{\mu_1(1-\rho_1)}\right]. \tag{18}$$

Thus, the average residence time of a C2 data packet is

$$\begin{aligned} T_{Q_2} &= Q_1 T_{Q_1} + \frac{Q_2}{\lambda_2} \\ &= \frac{\rho_1}{\mu_1(1-\rho_1)^2} + \frac{1}{\mu_2(1-\rho_1-\rho_2)}\left[1+\frac{\mu_2\rho_1}{\mu_1(1-\rho_1)}\right]. \end{aligned} \tag{19}$$

6. Simulation Experiments and Discussion

The proposed multitask schedule model can be used in many network applications. In coalmine, there are many monitoring and information management systems for its safety and production, which are the typical multitask wireless sensor network applications. In this kind of monitoring systems, the usual detecting period is 20 seconds and the number of monitoring nodes is usually more than 200. Thus, the proposed model applied in the gas warning system needs to process the data of thousands of sensor nodes. Moreover, the network delay and processing time need to be considered in practice applications.

In this experiment, we use the practical data from Huoerxinhe Coal Mine, China, which lay the gas warning wireless network with the same system structure as in Figure 1. In this network, the backbone network is optical fiber Ethernet, based on which network is partitioned into many zones. In each zone, a number of wireless sensor nodes are evenly laid out. Various monitoring data, such as gas concentration, CO concentration, CO_2 concentration, and so on are detected in real time by the sensor nodes. These data will be collected to the Sink node in the zone. Subsequently, all data are transferred to the server by the sink nodes in each zone. The transfer capability of Sink nodes is the bottleneck of the capability of the network system. In the test data set from Huoerxinhe Coal Mine, a Sink node is able to send 200 UDP packets per second, from which 90 UDP packets arrive at the target node. Each UDP packet contains 85 bytes.

The parameters λ, λ_1, λ_2, μ, μ_1, and μ_2 in formula (6), (15), and (19) are decided according to the field test.

If the priority processing rule is not employed, that is, the sink node employs the data processing model based on queue system without priority, the $\lambda = 90$ packets/s and $\mu = 200$/s. According to formula (6), the average delay of each packet is 9.1 ms. If the priority processing rule is employed, that is, the sink node employs the data processing model based on queue system with priority, the data are distinguished with different priorities. Taking the coal monitoring system as an example, the gas concentration and monitoring control command are with higher priority and others are with lower priority.

According to the statistics, the probability of C1 occurrence is 0.10 and the probability of C2 occurrence is 0.90. Meanwhile, $\mu_1 = \mu_2 = 200$ packets/s, $\lambda = 90$ packets/s, $\lambda_1 = 9$ packets/s, and $\lambda_2 = 81$ packets/s. Thus, according to formulas (15) and (19), the average delay of C1 packets is 5.2 ms and the average delay of C2 packets is 9.7 ms.

The theoretical analysis shows that compared with the data processing model based on queue system without priority, the average delay of data packets processed with the model based on queue system with priority is reduced up to 43%. However, the average delay of data packets without priority is slightly reduced only 6.6%.

For observing the queue and service process of data packets in network nodes with the proposed model, we use MatLab to simulate the model. The model parameters λ, λ_1, λ_2, μ, μ_1, and μ_2 are set in accordance with the theoretical analysis. The simulation results are shown in Figures 6, 7, 8, and 9, which show the same results with theoretical analysis. In fact, operation practice of multitask wireless sensor network in Huoerxinhe Coal Mine also confirmed our theoretical analysis and simulation experiments.

7. Conclusions

In this paper, two data processing models with and without priority are proposed for multitask wireless sensor networks. The proposed models are established from the M/M/1 queue model. The average delay theory of data packets based on the proposed models is also deduced. The practical data from Huoerxinhe Coal Mine are used for testing the performances of the proposed two models applied in the coal

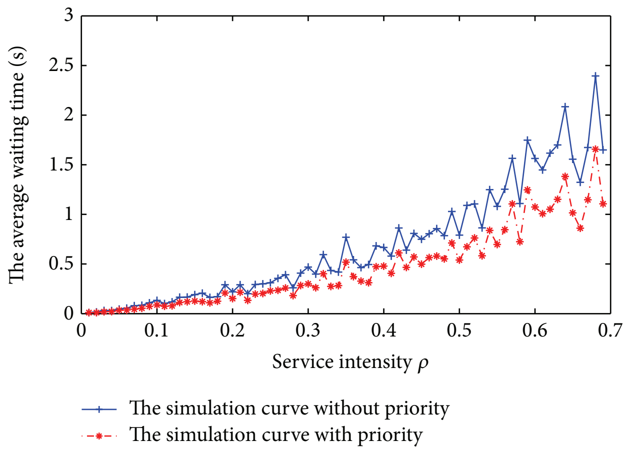


Figure 9: The curves of the average waiting time in the queuing models with priority and without priority.

safety monitoring system, which is a typical wireless sensor network application. The simulation results show that the average delay of data packets processed with the proposed model is significantly reduced. Compared with the average delay of data packets without priority, the proposed model can be applied to the multitask wireless sensor network harmonically.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgment

This work is supported by the National Natural Science Foundation of China 61379100 and 61001108, Bagui scholarship project; the Natural Science Foundation of Guangxi under Grant no. 2011GXNSFA018154 and 2012GXNSFGA060003; the Science and Technology Foundation of Guangxi under Grant no. 10169-1; Guangxi Scientific Research Project no. 201012MS274; Funded Projects of Innovation Plan for Guangxi Graduate Education no. gxun-chx2013t18; and Guangxi University for Nationalities Project no. 2012QD017.

References

[1] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "Wireless sensor networks: a survey," *Computer Networks*, vol. 38, no. 4, pp. 393–422, 2002.

[2] J. Yick, B. Mukherjee, and D. Ghosal, "Wireless sensor network survey," *Computer Networks*, vol. 52, no. 12, pp. 2292–2330, 2008.

[3] H.-S. Yin, Z. Liu, J.-S. Qian, K. Zhang, and J. Wu, "QoS model of multimedia integrated services digital network in coal mine," *Journal of China University of Mining and Technology*, vol. 39, no. 1, pp. 109–115, 2010.

[4] M. M. R. Mozumdar, L. Lavagno, and L. Vanzago, "A comparison of software platforms for wireless sensor networks: MANTIS, TinyOS, and ZigBee," *ACM Transactions on Embedded Computing Systems*, vol. 8, no. 2, pp. 123–129, 2009.

[5] C. Karlof and D. Wagner, "Secure routing in wireless sensor networks: attacks and countermeasures," *Ad Hoc Networks*, vol. 1, no. 2-3, pp. 293–315, 2003.

[6] J. Ager and L. Clare, "An integrated architecture for cooperative sensing networks," *Computer*, vol. 33, no. 5, pp. 106–108, 2000.

[7] N. Nasser, L. Karim, and T. Taleb, "Dynamic multilevel priority packet scheduling scheme for wireless sensor network," *IEEE Transactions on Wireless Communications*, vol. 12, no. 4, pp. 1448–1459, 2013.

[8] C. Duffy, U. Roedig, J. Herbert, and C. J. Sreenan, "Adding preemption to TinyOS," in *Proceedings of the 4th Workshop on Embedded Networked Sensors (EmNets '07)*, pp. 88–92, Cork, Ireland, June 2007.

[9] Y. Zhao, Q. Wang, W. Wang, D. Jiang, and Y. Liu, "Research on the priority-based soft real-time task scheduling in TinyOS," in *Proceedings of the International Conference on Information Technology and Computer Science (ITCS '09)*, pp. 562–565, Kiev, Ukraine, July 2009.

[10] M. Yu, S. Xiahou, and X. Y. Li, "A survey of studying on task scheduling mechanism for TinyOS," in *Proceedings of the 4th International Conference on Wireless Communications, Networking and Mobile Computing (WiCOM '08)*, pp. 1–4, Dalian, China, October 2008.

[11] K. Mizanian, R. Hajisheykhi, M. Baharloo, and A. H. Jahangir, "RACE: a real-time scheduling policy and communication architecture for large-scale wireless sensor networks," in *Proceedings of the 7th Annual Communication Networks and Services Research Conference (CNSR '09)*, pp. 458–460, Moncton, Canada, May 2009.

[12] J.-S. Qian, H.-S. Yin, X.-R. Liu, G. Hua, and Y.-G. Xu, "Data processing model of coalmine gas early-warning system," *Journal of China University of Mining and Technology*, vol. 17, no. 1, pp. 20–24, 2007.

[13] S. Asmussen, *Applied Probability and Queues*, Springer, New York, NY, USA, 2nd edition, 2003.

[14] L. Lipsky, *Queueing Theory: A Linear Algebraic Approach*, Springer, New York, NY, USA, 2nd edition, 2009.

[15] D. Gross and C. M. Harris, *Fundamentals of Queueing Theory*, John Wiley & Sons, New York, NY, USA, 1998.

[16] P. J. Smith, A. Firag, P. A. Dmochowski, and M. Shafi, "Analysis of the M/M/N/N queue with two types of arrival process: applications to future mobile radio systems," *Journal of Applied Mathematics*, vol. 2012, Article ID 123808, 14 pages, 2012.

Hindawi Publishing Corporation
Journal of Applied Mathematics
Volume 2014, Article ID 892832, 9 pages
http://dx.doi.org/10.1155/2014/892832

Research Article

Formal Proof of a Machine Closed Theorem in Coq

Hai Wan,[1] Anping He,[2] Zhiyang You,[3] and Xibin Zhao[1]

[1] *School of Software, Tsinghua University, NLIST, KLISS, Beijing 100084, China*
[2] *Guangxi Key Laboratory of Hybrid Computation and IC Design Analysis, Guangxi University for Nationalities, Nanning, Guangxi 530006, China*
[3] *General office of the Guangxi Zhuang Autonomous Region People's Government, Nanning, Guangxi 530003, China*

Correspondence should be addressed to Anping He; hapetis@gmail.com

Received 7 January 2014; Accepted 11 February 2014; Published 30 March 2014

Academic Editor: X. Song

The paper presents a formal proof of a machine closed theorem of TLA$^+$ in the theorem proving system Coq. A shallow embedding scheme is employed for the proof which is independent of concrete syntax. Fundamental concepts need to state that the machine closed theorems are addressed in the proof platform. A useful proof pattern of constructing a trace with desired properties is devised. A number of Coq reusable libraries are established.

1. Introduction

TLA$^+$ [1, 2] is a formal specification language for describing and reasoning about distributed and concurrent systems. It is based on mathematical logic, set theory, and linear time temporal logic TLA [3]. TLA$^+$ is widely used to write precise and formal specifications of discrete systems. The notion of machine closed plays an important role in the system specification. Generally speaking, a specification consists of two parts: one is a safety part and the other is a liveness part. The specification is called machine closed if the liveness part does not constrain the safety part. In [1], it is said that "*we seldom want to write a specification that isn't machine closed. If we do write one, it's usually by mistake*." Hence, we need to check whether the specification is machine closed. Fortunately, there is a well-known theorem, that is, machine closed theorem [4], stating that all the TLA$^+$ specifications are machine closed. More precisely, a TLA$^+$ specification which consists of a transition system and a possibly countable infinite fairness constraints is machine closed. Hence, in TLA$^+$ there is no need to verify the specification to see whether it is machine closed. In other words, TLA$^+$ specifications are constructed to be machine closed.

We are now working on formalizing a subset of TLA$^+$ in the theorem prover Coq. As an important part, we want to have the machine closed theorem in our formalization. There are mainly two ways to embed the theorem: one is to have it as an axiom and the other is to state it as a theorem and prove it. We think the second way is better. The reason is twofold:

(i) it needs to be very careful to introduce an axiom into a proof system, since sometimes the introduction of a new axiom may result in inconsistence;

(ii) it is worthwhile to have a formal proof of the theorem, though it is well-known. The proof will help to understand and check the previous formalizations (i.e., the fundamental definitions needed to state the theorem) and make the whole formalization more solid.

In this paper, we present a formal proof of a machine closed theorem in the theorem prover Coq. In order to do so, various fundamental definitions, such as traces, properties, safety property, liveness property, safety closure, and machine closed are given. Based on these definitions, the theorem is formally stated. The proof follows [4]. The key part of the proof is to find a strategy which can generate a trace with some desired properties. We designed a reusable proof pattern to guide this process. Besides this proof, the pattern is also used in proving whether a property is a liveness property; for example, both strong and weak fairness constraints are proved to be liveness properties using this pattern. The whole proof is done in a shallow-embedded manner, which makes the formalization independent of any concrete syntaxes. In other words, this work can be reused in other settings.

The rest of the paper is organized as follows. In Section 2, some preliminaries and a short introduction of TLA$^+$ are given. Section 3 presents the formalization of the machine closed theorem in Coq. The detailed proof is described in Section 4. Related work and concluding remarks are given in Section 5.

2. Preliminaries

2.1. TLA$^+$ Specifications. The TLA$^+$ specification is a formula of the form $Init \wedge \square[Next]_v \wedge L$, where v is a tuple usually containing all state variables in $Init$, $Next$, and L [4]. The first conjunct $Init$ describes the possible initial states of the system. The second conjunct of the specification asserts that every step (i.e., every pair of successive states in the system) either satisfies $Next$ or leaves the variables in v unchanged. Allowing such stuttering steps is a key ingredient to obtain compositionality of specifications. However, it also means that executions that stutter infinitely are allowed by the specification. The third conjunct L is a temporal formula stating the liveness constraints of the specification, and particularly it can be used to rule out infinite stuttering. The part $Init \wedge \square[Next]_v$ is known to be the safety part of the specification while L to be the liveness part. The machine closed theorem states that $(Init \wedge \square[Next]_v, L)$ is machine closed.

2.2. Definitions. We fix the set of states as St. St^* denotes the set of finite sequences of states, and St^ω is the set of infinite sequences of states. A sequence is also called a *trace* in this paper. The ith state in trace t is denoted by t_i. $|t|$ denotes the *length* of t, if t is finite.

A *property* is a set of infinite traces. Given a property P, we use $t \vDash P$, which means trace t satisfies property P, to denote $t \in P$. Given a finite trace $t \in St^*$, if $\exists t' \in St^\omega . t \circ t' \vDash P$ ($\circ$ is the traditional trace concatenation operator), then we say $t \vDash P$. $t_{[\cdots n]}$ and $t_{[n\cdots]}$ denote the prefix $(t_0 \cdots t_n)$ and suffix $(t_n t_{n+1} \cdots)$ of t, respectively. Following [5], a property P is a *safety property*, if $\forall t \in St^\omega . Pt \leftrightarrow \forall i . \exists t' \in St^\omega . t_{[\cdots i]} \circ t' \vDash P$. A property P is a *liveness property*, if $\forall t \in St^* . \exists t' \in St^\omega . t \circ t' \vDash P$.

An *action* a is a predicate over two states. Action a is a *subaction* of action a' if $\forall s\, s' . a\, s\, s' \rightarrow a'\, s\, s'$.

Following [6], we have the definitions of safety closure and machine closed. Given an arbitrary property P, its *safety closure* $C(P)$ is defined as $\{t \in St^\omega \mid \forall i . t_{[\cdots i]} \vDash P\}$. It can be proved that $C(P)$ is the smallest safety property containing P.

Definition. A pair of properties (S, L) is said to be *machine closed* if $C(S \cap L) = S$ and S is a safety property.

3. Formalization of Machine Closed Theorem

We work on the semantical level, in other words, we define all the concepts described in Section 2.2 in a shallow embedding manner. Throughout this section, we first describe the notions informally then give the corresponding Coq scripts.

3.1. Help Definitions. In the following scripts, we first define the set of state as St. A trace is a function of type $nat \rightarrow St$ (There are other ways to define traces. E.g., [7] uses coinductive data types to define infinite traces and [8] uses coinductive data types to define both infinite and finite traces. The reasons why we adapt the "function type" one is twofold: (i) the "function type" trace is intuitive; (ii) we only consider infinite traces, and we want to separate the finite traces and infinite traces at the data type level. In the subsequent section, finite traces are defined separately using inductive data type.). So given a trace t and a natural number i, $(t\, i)$ denotes the ith state of t (count from 0). $PredOn1$ and $PredOn2$ are the types of predicates over one state and two states, respectively. $PredOn1$ is the state predicate type while $PredOn2$ is the action type. A property is a set of traces, whose type is $Trace \rightarrow Prop$. $UptoN\, t\, t'\, n$ means traces t and t' are identical up to index n. $UptoN\_e\, t'\, t$ means traces t and t' are identical up to index $(n-1)$. $Offsettn$ returns $t_{[n\cdots]}$. $State1toProperty$ ($State2toProperty$, resp.) makes a trace predicate(i.e., a property) from a one-state (two-state, resp.) predicate. In order to make the representation more concise, we define three coercions which implicitly changes a one-state or a two-state predicate to properties, respectively.

```
Variable St: Set.
Definition Trace := nat -> St.
Definition PredOn1 := St -> Prop.
Definition PredOn2 := St -> St -> Prop.
Definition Property := Ensemble Trace.
Definition UptoN (t t': Trace) (n:nat):
Prop :=
  forall i, i <= n -> t i = t' i.
Definition UptoN_e (t t': Trace)
(n:nat): Prop :=
  forall i, i < n -> t i = t' i.
Definition OffsetN (t:Trace) (n:nat):
Trace :=
  fun i => t (n+i).
Definition State1toProperty (p:
PredOn1): Property :=
  fun t => p (t 0).
Definition State2toProperty (p:
PredOn2): Property :=
  fun t => p (t 0) (t 1).
Definition Actions := PredOn2.

Coercion State1toProperty: PredOn1 >->
Property.
Coercion State2toProperty: PredOn2 >->
Property.
Coercion Action2toProperty: Actions >->
Property.
```

In the following scripts, we have the traditional definitions of "finally," "always," "infinite often," "finally always," "enabled," "strong fairness," and "weak fairness."

```
Definition F (p: Property) (t:Trace):=
  exists n, p (OffsetN t n).
Definition G (p: Property) (t:Trace):=
  forall n, p (OffsetN t n).
Definition FG p := F (G p).
Definition GF p := G (F p).
Definition En (p2: PredOn2) (s: St) :=
exists s', p2 s s'.
Definition SF_Action (a: Actions):
Property :=
  fun t => GF (En a) t -> GF a t.
Definition WF_Action (a: Actions):
Property :=
  fun t => FG (En a) t -> GF a t.
```

3.2. Safety, Liveness, and Machine Closed. In the definitions of safety property, liveness property and safety closure, the notions of "finite trace" and "concatenation of a finite trace and an infinite trace" are used. Since we choose the function type to present traces, it is not very convenient to express finite traces. In order to avoid the notions of "finite trace" and "concatenation," we should change the definitions to their equivalent counterparts which only involve infinite traces.

Safety Property. We change "$\forall t \in S^{\omega}.Pt \leftrightarrow \forall i.\exists t' \in S^{\omega}.t_{[\cdots i]} \circ t' \vDash P$" to its equivalent formula "$\forall t \in S^{\omega}.Pt \leftrightarrow \forall i.\exists t' \in S^{\omega}.(UptoN\, t\, t'\, i) \wedge t' \vDash P$."

```
Definition IsSafetyProperty (p:
Property) :=
  forall t, p t <-> (forall i, exists
  t', UptoN t t' i /\ p t').
```

Liveness Property. We change "$\forall t \in S^{*}.\exists t' \in S^{\omega}.t \circ t' \vDash P$" to "$\forall t \in S^{\omega}.\forall i.\exists t' \in S^{\omega}.UptoN\_e\, t\, t'\, i \wedge t' \vDash P$" and have the following definition:

```
Definition IsLivenessProperty (p:
Property) :=
  forall t i, exists t', UptoN_e t t'
  i /\ p t'.
```

Safety Closure. "$\{t \in S^{\omega} \mid \forall i.t_{[\cdots i]} \vDash P\}$" is changed to "$\{t \in S^{\omega} \mid \forall i.\exists t'.UptoN\, t\, t'\, i \wedge t' \vDash P\}$".

```
Definition SafetyClosure (p: Property):
Property :=
  fun t => forall i, exists t', UptoN t
  t' i /\ p t'.
```

We use the notation mechanism of Coq to make the representations more succinct. $p\,[/\backslash]\,q$ denotes that the intersection of properties p and q. $p\,[<=]\,q$ expresses the fact that property p is a subset of property q. $p\,[=]\,q$ describes the fact that property p is equal to property q. $[C]p$ is the safety closure of p. $[S?]$ is a shorthand of predicate *IsSafetyProperty* while $[L?]$ is a shorthand of predicate *IsLivenessProperty*.

Machine Closed. It is defined as

```
Definition MachineClosed (s
p:Property): Prop :=
  [C] (s [/\] p) [=] s /\ [S?] s.
```

3.3. Transition System. As described in Section 2.1, the specification of a system is of form $Init \wedge \Box[Next]_v \wedge L$. Guided by this form, a transition system is parameterized by

(1) the set *St* of states,

```
Variable St: Set.
```

(2) predicates *Init* of type *PredOn*1*St* and *Next* of type *PredOn*2*St* characterizing the initial states and next-state relation, respectively, where we require that (`Parameter` is a synonym of `Variable` in Coq.): every state has a successor according to *Next*; in other words, *Next* is total.

```
Parameter Init: PredOn1 St.
Parameter Next: PredOn2 St.
Hypothesis next_input_enabled: forall
s, exists s', Next s s'.
```

(3) two sets of indexes: I and J. $I \cup J$ is finite or countable infinite and each $k \in I \cup J$, action a_k is a subaction of *Next*. Each $i \in I$ ($j \in J$, resp.), action a_i (a_j, resp.) is associated with a strong (weak, resp.) fairness constraint.

```
Parameter Acts_S: nat -> Actions St.
Parameter Acts_W: nat -> Actions St.
Definition Acts_SF := fun n =>
SF_Action _ (Acts_S n).
Definition Acts_WF := fun n =>
WF_Action _ (Acts_W n).
Hypothesis Acts_subaction_of_Next_SF:
  forall i s s', (Acts_S i) s s' ->
  Next s s'.
Hypothesis Acts_subaction_of_Next_WF:
  forall i s s', (Acts_W i) s s' ->
  Next s s'.
```

In the scripts, we use a function of type $nat \rightarrow Actions\ St$ to represent the set of actions. Note that the cases where the set of actions is finite or countable infinite are covered by the definitions of *Acts_S* and *Acts_W*. The essential point is that both *Acts_S* and *Acts_W* are of type $nat \rightarrow Actions\ St$. Suppose the set of I is finite, say its cardinality is N, we can build *Acts_S* as follows: for each $i < N$, map $Acts\_S\, i$ to the $(i+1)$th action in I; for each $i \geq N$, map $Acts\_S\, i$ to $(fun\, s\, s' \Rightarrow False)$. If the set of I is countable infinite, then for each $i \in \mathbb{N}$, map $Acts\_S\, i$ to the $(i+1)$th action.

Based on the above definitions, we can build the safety part *sp* and the liveness part *lp* for the transition system:

(i) *sp*: all traces allowed by *Init* and *Next*.

```
Definition sp :=
  fun (t:Trace _) => Init (t 0)/\
  forall n, Next (t n)(t (S n)).
```

(ii) *lp*: all traces allowed by the set of fairness constraints.

```
Definition lp: Property _ :=
  fun t => forall n, Acts_SF n t /\
  Acts_WF n t.
```

Finally, we get the theorem to prove

```
Theorem machine_closed:
  MachineClosed _ sp lp.
```

4. Proof of Machine Closed Theorem

The proof follows the proof of Proposition 4 in [4].

Given a specification of a transition system, $Init \wedge \Box Next \wedge L$ where $L = \forall i \in I.WF(a_i) \wedge \forall j \in J.SF(a_j)$, $\forall i \in I.a_i$ is a subaction of $Next$, $\forall j \in J.a_j$ is a subaction of $Next$ and $I \cup J$ is finite or countable infinite, we need to prove that $(Init \wedge \Box Next, L)$ is machine closed. Based on the discussion in Section 3.3, here we prove a more general case where both I and J are countable infinite. Hence, we can take I and J to be $\mathbb{N}$.

The whole proof is divided into two steps. First, we prove that a stronger version of the specification, in which all the weak fairness constraints are changed to their strong fairness counterparts, is machine closed. Second, we prove the original specification is machine closed.

4.1. The Stronger Specification. To build a stronger specification, first we need a mapping $f : nat \rightarrow Actions\ St$, which has the property that $ran(Acts\_S) \cup ran(Acts\_W) = ran(f)$, where $ran\ \bullet$ is the range of function $\bullet$. Informally speaking, through f we can get all the actions used in the original specification. In the following scripts, *even_odd_dec* is deployed to test whether a natural number is even and $div2\ n$ returns $n/2$.

```
Definition f (n:nat): Actions St :=
  match (even_odd_dec n) with
  |left _ => Acts_S (div2 n)
  |right _ => Acts_W (div2 n)
  end.
```

Then we can build the stronger specification that is equivalent to the original one except for

(i) the fairness constraint *lp* is replaced by *lp_stronger*. As we can see in the following scripts, each action is associated with a strong fairness constraint. It differs from the original one in which some actions are associated with weak fairness constraints, while the other with strong fairness constraints.

```
Definition lp_stronger := fun t =>
forall n, SF_Action _ (f n) t.
```

(ii) Based on assumptions *Acts_subaction_of_Next_SF* and *Acts_subaction_of_Next_WF*, the following theorem is proved:

```
Theorem Acts_subaction_of_Next:
  forall i s s', (f i) s s' ->
  Next s s'.
```

Since all the actions are indexed by a natural number and accessible through the number using f, in the sequel, we will use a natural number to represent the action: phrase "action i" is equivalent to "action $f(i)$."

The intermediate theorem about the stronger specification is

```
Theorem sp_lp_stronger_machine_closed:
  MachineClosed _ sp lp_stronger.
```

Following the definition of machine closed, we need to prove

(1) *sp* is a safety property;

(2) $C(sp \cap lp\_stronger) = sp$. There are two directions:

(1.1) $C(sp \cap lp\_stronger) \subseteq sp$;

(2.2) $sp \subseteq C(sp \cap lp\_stronger)$.

The set of all runs of the transition system is a safety property. $Init \wedge \Box[Next]$, which is equal to *sp*, defines a transition system. Hence, condition (1) is proved. Condition (2.1) can be proved from the fact that $sp \cap lp\_stronger \subseteq sp$ and the following theorem:

```
Theorem SafetyClosureSmallest:
  forall p p', p[<=]p' -> [S?] p' ->
  ([C] p) [<=] p'.
```

In order to prove condition (2.2), based on the definition of safety closure, it is sufficient to prove that for each $t \in sp$ and $n \in \mathbb{N}$, $t_{[\cdots n]}$ can be extended to an infinite trace t' such that $t' \in sp \wedge t' \in lp\_stronger$. As described in [4], given $t_{[\cdots n]}$ we need to construct a trace-generate strategy g which can generate a trace t' and t' have properties: $t'_{[\cdots n]} = t_{[\cdots n]} \wedge t' \in sp \wedge t' \in lp\_stronger$.

4.1.1. Trace-Generate Strategy. Roughly speaking, a trace-generate strategy is of type $S^* \rightarrow S$, which takes a finite trace as input and returns a state. In other words, a trace-generate strategy defines a scheduling policy which returns a next state based on the state sequence the system already produced. We first define the finite trace data type $FiniteTrace$ (*list* is a predefined type in Coq: Inductive list (A: Type): Type := nil: list A—cons: A $\rightarrow$ list A $\rightarrow$ list A. The concatenation of an element a of type A and a list l of type $list\ A$ is denoted by $a :: l$.).

```
Definition FiniteTrace := list St.
```

There are two ways to obtain a strategy function: (1) define it as a function of type *FiniteTrace* $\rightarrow$ *St* directly; (2) first define a relation of type *FiniteTrace* $\rightarrow$ *St* $\rightarrow$ *Prop* and then derive a function of type *FiniteTrace* $\rightarrow$ *St* from the relation using the classical choice axiom. In our case, we choose the second way, since we only care about the relation. And furthermore defining a relation is easier than defining a function: a function is difficult to define if the computation is complex. In our case, we essentially defined a scheduling policy, which is complex.

In order to use the classic choice axiom, the relation g should be total; that is, for each finite trace t there exists a state s such that $g\,t\,s$. We only consider this kind of strategy relations. The set of all valid strategy relations is captured by *VGenStrategy*.

```
Definition GenStrategy := FiniteTrace
-> St -> Prop.
Definition GenStrategyF := FiniteTrace
-> St.
Definition VGenStrategy :=
  {g:GenStrategy | (forall f, exists
  s:St, g f s)}.
```

A trace t can be derived based on a valid strategy relation *vsr*. The main derivation steps are as follows:

(1) a strategy function f is derived from *vsr* using the choice axiom;

(2) given a strategy function f and a natural number n, prefix $t_{[\cdots n]}$ is calculated recursively by *GSF_to_FiniteTrace*;

```
Fixpoint GSF_to_FiniteTrace
(sf:GenStrategyF)(n:nat) {struct n}
  : FiniteTrace :=
  match n with
  |0 => sf nil :: nil
  |S n' => let t' := GSF_to_FiniteTrace
  sf n' in
      sf t':: t'
  end.
```

(3) the prefix is always not nil, which means we can always get the last state t_n that is the nth state in t. (*myhead* of type $\forall l$: *FiniteTrace*.$l \neq nil$ $\rightarrow$ *St* returns the head of a finite trace. It requires a proof that the input finite trace is not nil. This fact is provided by theorem *GSF_to_FiniteTrace_notnil*.)

```
Definition GSF_to_Trace (g:
GenStrategyF): Trace :=
  fun n => myhead (GSF_to_FiniteTrace
  g n)
    (GSF_to_FiniteTrace_notnil g n).
```

Finally we get theorem *GenStrategy_to_Trace*, through which we can derive a trace t based on a valid strategy g and t fulfils our constraints that (nil, t_0) and $(t_{[\cdots i]}, t_{i+1})$ (for all $i \in \mathbb{N}$) satisfies the strategy relation. *GetG* is used to get the *GenStrategy* part from a *VGenStrategy* and *GetPrefixN* is used to get the first n states from a *Trace*.

```
Theorem GenStrategy_to_Trace:
  forall g: VGenStrategy, exists
  t:Trace,
    forall n, (GetG g) (GetPrefixN t n)
    (t n).
```

4.1.2. Design a Trace-Generate Strategy. Recall that we need to prove condition (2.2) in the previous subsection: given a trace $t \in sp$ and a position n, a trace t' should be constructed such that three properties (a) $t'_{[\cdots n]} = t_{[\cdots n]}$, (b) $t' \in sp$ and (c) $t' \in lp\_stronger$ are hold. To achieve the goal, firstly, we need to design a trace-generate strategy relation (the relation is designed with the three properties under consideration); secondly, we need to prove the relation is valid; at last, we need to prove the generated trace conforms to the properties. We fix t, n, ft, and s in the sequel.

Define the Relation. The design principles of the relation are as follows:

(1) at any point, the set of schedulable actions should be finite and all actions in the set is enabled;

(2) at any point, if the schedulable action set is not empty, the action that has not been executed for the longest time is scheduled to execute. In event of ties, the action with minimal index is chosen. Principle 1 is the key to this principle, since if the schedulable action set is infinite, there may exist an action which is infinitely enabled but not infinitely executed;

(3) at any point, if the schedulable actions set is empty, then pick *Next* to execute since *Next* is total;

(4) $t_{[\cdots n]}$ should be a prefix of the generated trace.

The strategy is defined as:

```
Definition MC_strategy (t:Trace _)
(n:nat) :=
  fun (ft:FiniteTrace St)(s:St) =>
    match ft with
    |nil => s = t 0
    |cons s' tl =>
      (0<length ft<=n /\t (length ft)=s)
      \/
      (n<length ft /\ (forall i,~
      MC_Enabled ft s' i)/\Next s' s)
      \/(n<length ft /\exists i,
      MC_Enabled ft s' i /\
      MC_theMin ft s' i /\ (Acts i) s' s)
  end.
```

Intuitively "$(MC\_strategy\, t\, n)\, ft\, s$" has the following implicit meanings:

(i) ft is the finite trace that the system has already produced;

(ii) if $(MC\_strategy\, t\, n)$ $ft\, s$ holds, then s is the next state the system will generate.

Depend on the length of ft, there are 4 cases of how the strategy generates a next state: (a) ft is nil; (b) $|ft| \le n$; (c) $|ft| > n$ and all the actions are not enabled; (d) $|ft| > n$ and there is an enabled action.

$MC\_theMin\, ft\, s'\, i$ denotes that the action i has the properties mentioned in the second principle. In order to define *MC_theMin*, there are several concepts that need to be represented: (a) given a finite trace and an action, the number of steps that the action continuously has not been executed up to now (*nsteps*); (b) notion of the largest number of nonexecuted steps (*MC_theLongestEnabled*); (c) the smallest index with the largest nonexecuted number *MC_theMin*. Given a finite trace ft, a nat i, and an action predicate a, $NoExeOfActionUpTo$ $ft\, i\, a$ returns the number of executable actions of ft.

```
Fixpoint nsteps (ft:FiniteTrace St)
(i:nat)
  (a: Actions St) {struct ft}: nat :=
  if lt_le_dec i (length ft) then
    match ft with
    |nil => 0
    |hd :: nil => 0
    |hd :: (hd':: _) as tail =>
      if action_dec a hd' hd then 0
      else S (NoExeOfActionUpTo tail
      i a)
    end
  else 0.
Definition MC_theLongestEnabled (ft:
FiniteTrace _) s idx :=
  MC_Enabled s idx /\

  (forall idx', MC_Enabled s idx' ->
    nsteps ft idx'(f idx')<=nsteps ft
    idx(f idx).
Definition MC_theMin (ft: FiniteTrace
_) s idx :=
  MC_theLongestEnabled ft s idx /\
    (forall idx', MC_theLongestEnabled
    ft s idx'->idx <= idx').
```

nsteps needs more descriptions. For its arguments, ft is the finite trace, i is the index of the action and a is the action. *lt_le_dec* returns *true* if and only if its first argument is less than the second argument. The first if-statement is the key to make the schedulable action set finite, but it does so in an implicit way, which will be explained in section "*Prove the Properties*." $action\_dec\, a\, hd'\, hd$ returns *true* if and only if $a\, hd\, hd'$ holds.

Useful Theorems. In the sequent proofs, we want to choose an element with a specific property from a set, for example, the minimal or the maximal element from a set according to some order. We use theorems to do so and these theorems are of a similar pattern: given a set, if the set is not empty, then there exists an element in the set with some specific properties.

(i) *Get the Minimal Element*. The following theorem states that given a set P, if P is not empty and R is a well-founded order, then there exists a minimal element in P according to order R.

```
Theorem ExistTheMin_general:
  forall (A:Set)(R: A -> A -> Prop)
  (P: Ensemble A),
    well_founded R ->(~ Empty_set _ P)
    ->
      (exists n, In _ P n /\ (forall i,
      In _ P i ->~ R i n)).
```

(ii) *Get the Maximal Element*. The following theorem expresses that given a set P of natural numbers, if there exists a natural number *max* which is larger than all the elements in P, then there exists a maximal element in P.

```
Theorem ExistTheMax_nat:
  (~ Empty_set _ P) ->
    (exists max, (forall n, In _ P
    n -> n <= max)) ->
      exists m, In _ P m /\ forall n,
      In _ P n -> n <= m.
```

These theorems can be thought as a kind of element choosers, which can be used to pick up a specific element from a set. Combining these choosers with proper initializations of the set predicates (i.e., P in the theorem), we can obtain some useful complex choosers, such as the one choosing the smallest index among the indexes whose associated actions have not been executed for the longest time.

(iii) The following theorem represents that given P, a finite set of natural numbers, and F, a relation over two natural numbers, if F is total on P (i.e., for each element in P there exists a number F-related to it), and F is transitive over the second parameter (i.e., for each element in P, if a number is F-related to the element then any number larger than the number is also F-related to the element) then there exists an m such that m is F-related to each element in P.

```
Theorem PFiniteExistsAMax:
  forall (P: Ensemble nat)
  (F: nat->nat->Prop),
    Finite _ P->(forall i, In _ P i ->
    exists xi, F i xi) ->
      (forall i xi xi', In _ P i->F i
      xi->xi<=xi'->F i xi') ->
        exists m, forall i, In _ P i ->
        F i m.
```

Prove the Validity. The validity of *MC_strategy* is demonstrated by the following theorem:

```
Theorem MC_strategy_Valid:
  forall t n f, exists s:St, MC_strategy
  t n f s.
```

The theorem is proved by case analysis, which corresponds to the four cases of *MC_strategy*. For the first 3 cases, they are not hard to prove. For the last case, we need to prove the following theorem:

```
Theorem MC_theMin_Valid:
  forall ft s, (exists i, MC_Enabled
  ft s i) ->
    exists idx, MC_theMin ft s idx.
```

This theorem is proved based on the theorems described in the previous subsection.

Prove the Properties. At this point, we have a valid trace-generate relation, based on which we obtain a generated trace. We need to prove that the generated trace conforms to the three properties.

Recall that t' is the generated trace. Property (a) (i.e., $t'_{[\cdots n]} = t_{[\cdots n]}$) is ensured by the first and second cases of *MC_strategy*: if ft is nil, then the next state is t_0; else if $|ft| \leq n$, then the next state is $t_{|ft|}$. Property (b) is proved based on the facts that: (1) $t \in sp$; (2) case 3 generates a next state based on *Next*; (3) case 4 generates a state based on an action which is a subaction of *Next*. The last property (c) expresses that for each action if it is infinitely enabled in trace t', then it is also infinitely executed. We prove this by contradiction:

(1) if there is an action that is infinitely enabled but finitely executed, then there is a set of actions that are infinitely enabled but only finitely executed;

(2) properly pick an action a from the set and a position n such that a is not executed in the suffix $t'_{[n\cdots]}$;

(3) pick a position $m > n$ such that at m the next action chosen to execute by the scheduler is a, which conflicts with step 2.

In order to make a concise representation, we have the following definitions:

```
Definition NotAfter t a n :=
  forall i, n<=i -> ~a t[i] t[S i].
Definition InfEn t a :=
  GF (En a) t.
```

For the second step, corresponding to *MC_strategy*, we choose action idx and n such that (n, idx) is the smallest element in $\{(n, idx) \mid idx <= n + 1 \wedge InfEn\, t'(f\, i) \wedge NotAfter\, t'(f\, i)\, n\}$ (the order between two pairs $p_1 = (a_1, b_1)$ and $p_2 = (a_2, b_2)$ is the classical lexicographic order: $p_1 < p_2 \triangleq a_1 < a_2 \vee (a_1 = a_2 \wedge b_1 < b_2)$), that is, the smallest idx with the smallest n. The first conjunct is used to ensure that we only consider the actions whose indexes are less than the length of ft. This constraint corresponds to the first if-statement in *nsteps*. The second conjunct guarantees that there always is an arbitrary large position at which the action is enabled. It can be proved that the set is not empty based on the assumption in step 1. Again, there is a chooser theorem for this operation.

Intuitively there is some point after n where action idx is the next action to execute, because it has not been executed since n and its index is the smallest. Now we need to find such a point m at which:

(1) for each action $idx' \leq n$, one of the following cases holds:

  (a) if idx' is infinitely enabled and infinitely executed, then $nsteps\, t_{[\cdots m]} idx' < nsteps\, t_{[\cdots m]} idx$;

  (b) if idx' is infinitely enabled but finitely executed, then

    (i) if action idx' is executed at least once after n, then $nsteps\, t_{[\cdots m]} idx' < nsteps\, t_{[\cdots m]} idx$;

    (ii) if action idx' is not executed after n, then $nsteps\, t_{[\cdots m]} idx' \leq nsteps\, t_{[\cdots m]} idx$;

  (c) if idx' is finitely enabled, then idx' is not enabled at m;

(2) for each action $idx' > n$, $nsteps\, t_{[\cdots m]} idx' \leq nsteps\, t_{[\cdots m]} idx$ and $idx < idx'$;

Suppose we find m satisfying both conditions. The theorem *PropertyOfM* (we omit the preconditions, since they are too many) states that idx is the smallest index with the longest nonexecute steps (i.e., $MC\_theMin\, t'[\cdots m] t'[m]\, idx$), hence only the fourth case of *MC_strategy* can be true. Based on the uniqueness of *MC_theMin*, we can infer that action idx is the action the scheduler chooses to execute at m.

```
Lemma PropertyOfM:
  ··· ->
    MC_theMin t' [···m] t' [m] idx.
Theorem MC_theMin_Unique:
  forall ft s i i', MC_theMin ft
  s i -> MC_theMin ft s i' ->
  i=i'.
```

In order to obtain such m, we first construct that m' s.t. condition (1) holds and then construct that m s.t. both conditions hold. We use theorem *PFiniteExistsAMax* to get m'. In the theorem P is the set of indexes less than $(n+1)$ and F is defined as

```
Definition F := fun i xi =>
  let inf_en := (GF St (En St (Acts i))
  t') in
```

```
let inf_exe := (GF St (Acts i) t') in
let tp := (GetPrefixN St t' (S xi)) in
let a := Acts idx in
let a' := Acts i in
n <= xi /\
((inf_en /\ inf_exe /\ nstep tp i a' <
nstep tp idx a) \/
(inf_en /\~ inf_exe /\
(((exists n, min_n <= n /\ a' (t' n)
(t' (S n)))/\
  nstep tp i a' < nstep tp idx a) \/
((forall n, min_n <= n ->~a' (t' n)
(t' (S n)))/\
  nstep tp i a' <= nstep tp idx a)))
  \/
(~ inf_en /\ (forall k, xi <= k -> ~
En St a' (t' k)))).
```

The second conjunct consists of four disjuncts, each of which corresponds to a sub condition in condition (1). The total and transitive properties of F are proved by case analysis. By *PFiniteExistsAMax* we obtain the m' which is F-related to all the actions whose indexes are less than $(n + 1)$. For each index $i > n$, by using theorem *greater_less* we know $nstep\, t_{[\cdots m]}\, i\, (f\, i) \leq (m + 1 - i)$, and by theorem *less_greater*, we know $m - n \leq nstep\, t_{[\cdots m]}\, idx\, (f\, idx)$; hence, we know m also holds for condition (2). Thus m is the position we want—property (c) holds on trace t'.

```
Lemma greater_less:
  forall ft i a, nstep ft i a <= length
  ft - i.
Lemma less_greater:
  forall t idx a n, idx <=S n ->
  (forall j, n<=j -> ~a (t j)
  (t (S j))) ->
    forall k, n < k -> k-n<=nstep
      (GetPrefixN St t (S k)) idx a.
```

Finally, we prove that the stronger specification is machine closed.

4.2. The Original Specification. Based on theorem *sp_lp_stronger_machine_closed*, we need to prove theorem *machine_closed*. According to the definition of machine closed, the proof is sketched as

(1) prove sp is a safety property;

(2) prove $C(sp \cap lp) = sp$. There are two directions:

(a) $C(sp \cap lp) \subseteq sp$. The proof is similar to the proof of condition (2.1) in Section 4.1;

(b) $sp \subseteq C(sp \cap lp)$. Given t and i, by theorem *sp_lp_stronger_machine_closed* we can get an extended trace t_0 of $t_{[\cdots i]}$ such that $t_0 \in sp \wedge t_0 \in lp\_stronger$. Hence, the only subgoal needed to solve is to prove that t_0 is also in lp. It is sufficient to prove that

(i) for each $i \in I$, t_0 satisfies the strong fairness constraint of action a_i—this holds, since t_0 satisfies the strong fairness constraint of action $a_{f(2*i)}$ which is equal to a_i.

(ii) for each $i \in J$, t_0 satisfies the weak fairness constraint of action a_j—this holds, since t_0 satisfies the strong fairness constraint of action $a_{f(2*j+1)}$ which is equal to a_j and the following theorem which expresses that if a trace satisfies the strong fairness constraint of an action it also satisfies the weak fairness constraint of that action:

```
Theorem SF_imp_WF:
  forall (a:Actions), SF_Action
  a [->] WF_Action a.
```

5. Related Works and Concluding Remarks

The machine closed theorem is first proved in [4]. There is already some work that embeds TLAi n a theorem prover [9], but to our best knowledge, this is the first time that the theorem is formally proved in a theorem prover. There are several other works that concern the definitions of properties. These works can be discussed in two steps. The first step is how traces are represented. In [10], a function of type $nat \rightarrow St$ is chosen, which is the same as our solution. In other works, inductive and/or coinductive types are used [7, 8, 11]. In [12], the authors propose a more general solution, in which they do not commit to a particular formalization of traces; instead, they exploit the module system of Coq and only list the interface of traces. The second step is how the safety and liveness properties are defined. In [7], the safety property is defined as a state invariant of a transition system and the liveness property is not defined formally.

In this paper, we present a formal proof of the the machine closed theorem in theorem prover Coq. Various fundamental definitions, such as traces, properties, safety property, liveness property, safety closure and machine closed, are given. Based on these definitions, the theorem is formally stated and proved. The main proof of machine closed theorem is done using the section mechanism of Coq. This mechanism makes our formalization adaptable. It can be encapsulated into a module or a record. In our case study, we used the module type. The result module is general, since it is at the semantics level (because we do the proof in a shallow embedding manner) and thus is independent of any concrete syntax. This work also results in several reusable Coq libraries. The Coq scripts can be provided upon request.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

The paper was supported in part by the 973 Program of China (Grant no. 2010CB328000), the National Natural Science Foundation of China (Grants nos. U1201251 and 61133016), the National 863 Plan of China (Grant no. 2012AA040906), National Natural Science Foundation of Guangxi (Grant no. 2013GXNSFAA019342), GUN Project no. 2012Q017 and the Bagui scholarship Project of Guangxi.

References

[1] L. Lamport, "Specifying Systems," Addison-Wesley, 2002, http://research.microsoft.com/users/lamport/tla/tla.html.

[2] S. Merz, "The specification language TLA^+," in *Logics of Specification Languages*, pp. 401–451, Springer, Berlin, Gemany, 2008.

[3] L. Lamport, "Temporal logic of actions," *ACM Transactions on Programming Languages and Systems*, vol. 16, no. 3, pp. 872–923, 1994.

[4] M. Abadi and L. Lamport, "Old-fashioned recipe for real time," *ACM Transactions on Programming Languages and Systems*, vol. 16, no. 5, pp. 1543–1571, 1994.

[5] B. Alpern and F. B. Schneider, "Defining liveness," Tech. Rep., Ithaca, NY, USA, 1984.

[6] M. Abadi and L. Lamport, "The existence of refinement mappings," *Theoretical Computer Science*, vol. 82, no. 2, pp. 253–284, 1991.

[7] S. Coupet-Grimal, "An axiomatization of linear temporal logic in the calculus of inductive constructions," *Journal of Logic and Computation*, vol. 13, no. 6, pp. 801–813, 2003.

[8] Y. Bertot and P. Castéran, *Interactive Theorem Proving and Program Development*, Coq'Art: The Calculus of Inductive Constructions, Springer, Berlin, Germany, 2004.

[9] S. Merz, "Yet another encoding of TLA in Isabelle," Rapport de Recherche, Institut für Informatik, TU München, Germany, 1997.

[10] O. Müller and T. Nipkow, "Combining model checking and deduction for I/O-automata," in *Tools and Algorithms For the Construction and Analysis of Systems*, pp. 1–16, Springer, 1995.

[11] O. Müller and T. Nipkow, "Traces of I/O automata in Isabelle/HOLCF," in *TAPSOFT'97: Theory and Practice of Software Development*, M. Bidoit and M. Dauchet, Eds., vol. 1214, pp. 580–594, 1997.

[12] M. H. Tsai and B. Y. Wang, "Formalization of CTL^* in calculus of inductive constructions," in *ASIAN*, pp. 316–330, 2006.