First-Order Indefinability of Answer Set Programs on Finite Structures

Yin Chen
Department of Computer Science
South China Normal University, China
Email: gzchenyin@gmail.com

Yan Zhang and Yi Zhou
School of Computing and Mathematics
University of Western Sydney, Australia
Email: {yan,yzhou}@scm.uws.edu.au

Abstract

An answer set program with variables is first-order definable on finite structures if the set of its finite answer sets can be captured by a first-order sentence, otherwise this program is first-order indefinable on finite structures. In this paper, we study the problem of first-order indefinability of answer set programs. We provide an Ehrenfeucht-Fraïssé game-theoretic characterization for the first-order indefinability of answer set programs on finite structures. As an application of this approach, we show that the well-known finding Hamiltonian cycles program is not first-order definable on finite structures. We then define two notions named the 0-1 property and unbounded cycles or paths under the answer set semantics, from which we develop two sufficient conditions that may be effectively used in proving a program's first-order indefinability on finite structures under certain circumstances.

Introduction

Answer Set Programming (ASP) is an important programming paradigm for declarative problem solving. In recent years, it has demonstrated profound applications in many areas such as semantic web, robotic planning and bioinformatics. Recent work on ASP has extended the traditional ASP framework by allowing variables in program rules, which we call first-order ASP, while the semantics of first-order ASP is defined via second-order logic (Ferraris, Lee, & Lifschitz 2010; Lin & Zhou 2007). Consequently, such extended answer set programs have significantly increased the expressive power compared to propositional answer set programs (Baral 2003).

Nevertheless, computing first-order answer set programs is difficult due to their inherited second-order logic semantics. One related issue is the first-order definability (indefinability) problem. An answer set program with variables is first-order definable on finite structures if the set of its finite answer sets can be captured by a first-order sentence, otherwise it is first-order indefinable on finite structures. Since most of our applications on ASP focus on finite structures, results about the first-order definability on finite structures, both positive and negative, will have important impacts to current ASP research. First, results in this aspect will provide a theoretic foundation to characterize the expressiveness of first-order ASP and hence to establish its close connections to classical first-order and second-order logics. It is observed that the first-order ASP generalizes traditional Datalog, and the expressive power and complexity of datalog programs have been well studied (Dantsin & et al 2001). As such, major first-order (in)definability results in Datalog, e.g. (Ajtai & Gurevich 1994; Cosmadakis 1989), may be carried over to first-order ASP. Nevertheless, such results are generally not applicable in proving a program's first-order (in)definability under our context, not only because of the additional consideration of negation as failure under answer set semantics, but also due to the fact that these results are mainly semantic characterizations on datalog programs and queries.

On the other hand, as evident from previous research from Datalog and finite model theory (Cosmadakis 1989; Ebbinghaus & Flum 1999), exploring first-order definability for a problem like this is a challenging task. In order to obtain certain results, especially negative results, very often new concepts and techniques have to be developed, which may also be useful for other related research.

In this paper, we focus on the negative results of first-order definability of answer set programs on finite structures. We first provide an Ehrenfeucht-Fraïssé game-theoretic characterization for the first-order indefinability of answer set programs. Using this approach, we show that the well-known finding Hamiltonian cycles program is not first-order definable. We then further propose new notions named the 0-1 property and unbounded cycles or paths under answer set semantics, from which we develop two sufficient conditions that may be effectively used in proving a program's first-order indefinability on finite structures under certain circumstances.

Basic Concepts and Definitions

We consider a second-order language with equality but without function symbols. A *vocabulary* is a finite set that consists of *constant symbols* and *relation symbols* including equality $=$. We denote the sets of constant symbols of a vocabulary τ by $\mathcal{C}(\tau)$ and relation symbols by $\mathcal{R}(\tau)$ respectively. Given a vocabulary, *terms*, *atoms*, (first-order or second-order) *formulas* and *sentences* are defined as usual. An atom is called an *equality* atom if it is of the form $t_1 = t_2$, where t_1 and t_2 are terms, and a *proper* atom other-

Copyright © 2010, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

wise.

A *finite structure* $\mathcal{A}$ of vocabulary τ is a tuple $(A, c_1^{\mathcal{A}}, \cdots, c_m^{\mathcal{A}}, R_1^{\mathcal{A}}, \cdots, R_n^{\mathcal{A}})$, where A is a finite set called the *domain* of $\mathcal{A}$, each $c_i^{\mathcal{A}}$ $(i = 1, \cdots, m)$ is an element in A which corresponds to a constant symbol c_i in $\mathcal{C}(\tau)$, and each $R_i^{\mathcal{A}}$ $(i = 1, \cdots, n)$ is a k-ary relation on A which corresponds to a k-ary relation symbol R_i in $\mathcal{R}(\tau)$. Sometimes, we also use $\mathsf{Dom}(\mathcal{A})$ to denote the domain of structure $\mathcal{A}$. In this paper, we will only consider finite structures in our context.

Given two vocabularies τ_1 and τ_2 where $\tau_2 \subseteq \tau_1$, and a finite structure $\mathcal{A}$ of τ_1, we say that the *restriction of $\mathcal{A}$ on* τ_2, denoted by $\mathcal{A}|\tau_2$, is a structure of τ_2 which has the same domain of $\mathcal{A}$, and for each constant c and relation symbol R in τ_2, $c^{\mathcal{A}}$ and $R^{\mathcal{A}}$ are in $\mathcal{A}|\tau_2$. On the other hand, if we are given a structure $\mathcal{A}'$ of τ_2, a structure $\mathcal{A}$ of τ_1 is an *expansion of $\mathcal{A}'$ to* τ_1, if $\mathcal{A}$ has the same domain of $\mathcal{A}'$ and retains all $c^{\mathcal{A}'}$ and $R^{\mathcal{A}'}$ for all constants c and relation symbols R in τ_2.

Let $\mathcal{A}$ be a structure. We usually write a tuple $(t_1, \cdots, t_n)$ as the form $\overline{t}$, where $\{t_1, \cdots, t_n\}$ is either a set of terms or a set of elements from $\mathsf{Dom}(\mathcal{A})$. If $\overline{a} = (a_1, \cdots, a_s)$ is a tuple of elements from $\mathsf{Dom}(\mathcal{A})$, i.e. $a_i \in \mathsf{Dom}(\mathcal{A})$ $(1 \leq i \leq s)$, then we simply write $\overline{a} \in \mathsf{Dom}(\mathcal{A})^s$.

For two tuples $\overline{t} = (t_1, \cdots, t_m)$ and $\overline{t'} = (t'_1, \cdots, t'_n)$, we may simply write $\overline{t'} \subseteq \overline{t}$ if $\{t'_1, \cdots, t'_n\} \subseteq \{t_1, \cdots, t_m\}$.

Consider a structure $\mathcal{A} = (A, c_1^{\mathcal{A}}, \cdots, c_m^{\mathcal{A}}, R_1^{\mathcal{A}}, \cdots, R_n^{\mathcal{A}})$ and $S \subseteq A$ where $\{c_1^{\mathcal{A}}, \cdots, c_m^{\mathcal{A}}\} \subseteq S$. Structure $\mathcal{A} \restriction S$ is called a *substructure* of $\mathcal{A}$ *generated* from S, if $\mathcal{A} \restriction S = (S, c_1^{\mathcal{A}}, \cdots, c_m^{\mathcal{A}}, R_1^{\mathcal{A}\restriction S}, \cdots, R_n^{\mathcal{A}\restriction S})$, where for any tuple $\overline{a}$ from S, $\overline{a} \in R_i^{\mathcal{A}\restriction S}$ iff $\overline{a} \in R_i^{\mathcal{A}}$ $(1 \leq i \leq n)$.

The *quantifier rank* $qr(\varphi)$ of a first-order formula φ is the maximum number of nested quantifiers occurring in φ: $qr(\varphi) = 0$ if φ is atomic, $qr(\varphi_1 \vee \varphi_2) = qr(\varphi_1 \wedge \varphi_2) = max(qr(\varphi_1), qr(\varphi_2))$, $qr(\neg\varphi) = qr(\varphi)$, and $qr(\exists x\varphi) = qr(\forall x\varphi) = qr(\varphi) + 1$.

With a fixed vocabulary τ, we consider two finite structures $\mathcal{A}$ and $\mathcal{B}$, and $m \in \mathbb{N}$. $\mathcal{A}$ and $\mathcal{B}$ are *m-equivalent*, denoted by $\mathcal{A} \equiv_m \mathcal{B}$, if for any first-order sentence φ with $qr(\varphi) \leq m$, $\mathcal{A} \models \varphi$ iff $\mathcal{B} \models \varphi$. $\mathcal{A}$ and $\mathcal{B}$ are called *isomorphic*, denoted as $\mathcal{A} \cong \mathcal{B}$, if there is a one-to-one and onto mapping h: $\mathsf{Dom}(\mathcal{A}) \to \mathsf{Dom}(\mathcal{B})$ such that for every constant $c \in \tau$, $h(c^{\mathcal{A}}) = c^{\mathcal{B}}$, and for every relation symbol $R \in \tau$ and every tuple $\overline{a}$ from $\mathsf{Dom}(\mathcal{A})$, $\overline{a} \in R^{\mathcal{A}}$ iff $h(\overline{a}) \in R^{\mathcal{B}}$.

If φ is a first-order or second-first sentence, we use $\mathsf{Mod}(\varphi)$ to denote the collection of all finite structures that satisfy φ. Let D be a finite set. We use $\mathsf{Mod}(\varphi)|D$ to denote the collection of all finite structures that satisfy φ *and* whose domains are D.

First-order Answer Set Programs

Syntax and semantics

A *rule* is of the form:

$$a \leftarrow b_1, \cdots, b_k, \mathsf{not}\ c_1, \cdots, \mathsf{not}\ c_l, \tag{1}$$

where a is a proper atom or the falsity $\bot$, and $b_1, \cdots, b_k, c_1, \cdots, c_l$ $(k, l \geq 0)$ are atoms. Here a is called the *head*, $\{b_1, \cdots, b_k\}$ the *positive body* and $\{\mathsf{not}\ c_1, \cdots, \mathsf{not}\ c_l\}$ the *negative body* of the rule respectively.

A *(first-order) answer set program* (or simply called *program*) Π is a finite set of rules. Every relation symbol occurring in the head of some rule of Π is called an *intentional predicate*, and all other relation symbols in Π are *extensional predicates*. The extensional predicates and individual constants occurring in Π form the extensional vocabulary of Π. We use notions $\tau(\Pi)$ to denote the vocabulary containing all of relation symbols and constants in Π, $\tau_{int}(\Pi)$ the vocabulary containing all intentional predicates in Π, and $\tau_{ext}(\Pi)$ the vocabulary containing all extensional predicates and constants in Π. We also use notions $\mathcal{P}(\Pi)$, $\mathcal{P}_{int}(\Pi)$ and $\mathcal{P}_{ext}(\Pi)$ to denote the sets all predicates, intentional and extensional predicates in Π respectively. A proper atom $P(\overline{t})$ is extensional (intentional) if P is extensional (intentional).

Sometimes, we simply call a relation $R^{\mathcal{A}}$ in a structure $\mathcal{A}$ an *intentional (extensional) relation* if $R^{\mathcal{A}}$ is the interpretation of an intentional (extensional, resp.) predicate of the underlying program Π.

Now we present the semantics of first-order answer set programs, which is a simplified version of the general stable model semantics (Ferraris, Lee, & Lifschitz 2010). For each rule r of form (1), we use $\widehat{r}$ to denote the sentence

$$\forall\overline{x}(\widehat{Body_r} \supset a),$$

where $\overline{x}$ is the tuple of all variables occurring in r, and $\widehat{Body_r}$ the formula $b_1 \wedge \cdots \wedge b_k \wedge \neg c_1 \wedge \cdots \wedge \neg c_l$. Given a rule r, by $\widehat{\Pi}$, we denote the sentence $\wedge_{r\in\Pi}\widehat{r}$.

Let $\mathcal{P} = \{P_1, \cdots, P_k\}$ and $\mathcal{P}' = \{P'_1, \cdots, P'_k\}$ be two sets of relation symbols where P_i and P'_i are of the same arity. By $\widehat{r}[+\mathcal{P}/\mathcal{P}']$, we mean the formula that is obtained from $\widehat{r}$ by replacing each relation symbol in $\mathcal{P}$ occurring in the head and positive body of r by the corresponding relation symbol in $\mathcal{P}'$. For instance, if r is a rule $R(x) \leftarrow P(x), \mathtt{not}\ Q(x)$, then $\widehat{r}[+\{Q, R\}/\{Q', R'\}] \equiv \forall x((P(x) \wedge \neg Q(x) \supset R'(x))$. We define $\widehat{\Pi}[+\mathcal{P}/\mathcal{P}'] = \wedge_{r\in\Pi}\widehat{r}[+\mathcal{P}/\mathcal{P}']$. Let P and Q be two predicate symbols or variables of the same arity. $P \leq Q$ stands for the formula $\forall\overline{x}(P(\overline{x}) \supset Q(\overline{x}))$. For the given $\mathcal{P} = \{P_1, \cdots, P_k\}$ and $\mathcal{P}' = \{P'_1, \cdots, P'_k\}$ where all P_i and P'_i have the same arity, $\mathcal{P} \leq \mathcal{P}'$ stands for formula $\bigwedge_{i=1}^{k} P_i \leq P'_i$, and $\mathcal{P} < \mathcal{P}'$ stands for formula $\mathcal{P} \leq \mathcal{P}' \wedge \neg(\mathcal{P}' \leq \mathcal{P})$.

Consider two vocabularies τ_1 and τ_2 where $\tau_2 \subseteq \tau_1$. Let ψ be a first-order or second-order sentence on τ_1 and $\mathcal{A}$ a finite structure of τ_2. We specify $\mathsf{Mod}(\psi)^{\mathcal{A}}_{\tau_1}$ as follows:

$$\mathsf{Mod}(\psi)^{\mathcal{A}}_{\tau_1} = \{\mathcal{A}' \mid \mathcal{A}' \in \mathsf{Mod}(\psi) \text{ and } \mathcal{A}' \text{ is an expansion of } \mathcal{A} \text{ to } \tau_1\}.$$

Definition 1 *(**Answer set program semantics**) Given a first-order answer set program Π and a structure $\mathcal{A}$ of $\tau_{ext}(\Pi)$. A structure $\mathcal{A}'$ of $\tau(\Pi)$ is an* answer set of Π based on $\mathcal{A}$ *iff $\mathcal{A}' \in \mathsf{Mod}(\psi)^{\mathcal{A}}_{\tau(\Pi)}$, where ψ is $\widehat{\Pi} \wedge \neg\exists\mathcal{P}^*(\mathcal{P}^* < \mathcal{P}_{int}(\Pi) \wedge \widehat{\Pi}[+\mathcal{P}_{int}(\Pi)/\mathcal{P}^*])$. We also use $\Im(\Pi, \mathcal{A})$ to denote the collection of all answer sets of Π based on $\mathcal{A}$. A structure $\mathcal{A}'$ of $\tau(\Pi)$ is an* answer set *of Π if there is some structure $\mathcal{A}$ of $\tau_{ext}(\Pi)$ such that $\mathcal{A}' \in \Im(\Pi, \mathcal{A})$.*

In Definition 1, minimization applies on intentional predicates while extensional predicates are viewed as the initial input of the program. Definition 1 is a simplified version of the general stable model semantics, where first-order sentences are allowed in a program and any set of predicates in the program may also be specified as intentional (Ferraris, Lee, & Lifschitz 2010).

First-order definability for answer set programs

Now we are ready to present a formal definition of first-order definability for an answer set program.

Definition 2 *(**First-order definability**) A program Π is called* first-order definable *iff there exists a first-order sentence ψ on vocabulary $\tau(\Pi)$ such that for every structure $\mathcal{A}$ of $\tau_{ext}(\Pi)$, $\mathsf{Mod}(\psi)^{\mathcal{A}}_{\tau(\Pi)} = \Im(\Pi, \mathcal{A})$. In this case, we say that ψ* defines *Π.*

Consider the program $\Pi = \{P(x) \leftarrow Q(x), \mathtt{not}\ R(x)\}$. According to Definition 2, Π can be defined by the sentence $\forall x(P(x) \equiv (Q(x) \wedge \neg R(x)))$.

Ehrenfeucht-Fraïssé Games for First-order Answer Set Programs

In this section we extend the traditional Ehrenfeucht-Fraïssé game-theoretic approach in finite model theory (Ebbinghaus & Flum 1999) to the context of answer set programs so that this approach may be used as a tool to prove the first-order indefinability for a given program.

Given two τ-structures $\mathcal{A} = (A, c_1^{\mathcal{A}}, \cdots, c_m^{\mathcal{A}}, R_1^{\mathcal{A}}, \cdots, R_n^{\mathcal{A}})$ and $\mathcal{B} = (B, c_1^{\mathcal{B}}, \cdots, c_m^{\mathcal{B}}, R_1^{\mathcal{B}}, \cdots, R_n^{\mathcal{B}})$, and $\overline{a} \in A^s$ and $\overline{b} \in B^s$, an *Ehrenfeucht-Fraïssé game*, which is played on $(\mathcal{A}, \overline{a})$ and $(\mathcal{B}, \overline{b})$, is played by two players named *spoiler* and *duplicator*. Each round of the game spoiler starts by picking an element from either A or B, and duplicator responds by picking an element from the opposite domain. For $k \geq 0$, let e_k (or f_k) be the element of A (or B resp.) at round k. By default, we denote e_{k+i} (or f_{k+i}) to be constant c_i's interpretation in $\mathcal{A}$ (or $\mathcal{B}$ resp.) where $i = 1, \cdots, m$. We say that duplicator *wins* round k ($k \geq 0$) iff the following conditions hold:

1. there is a bijective map h: $\overline{ae} \mapsto \overline{bf}$, where $h(\overline{a}) = \overline{b}$, $h(\overline{e}) = \overline{f}$, $\overline{e} = (e_1, \cdots, e_k, e_{k+1}, \cdots, e_{k+m})$ and $\overline{f} = (f_1, \cdots, f_k, f_{k+1}, \cdots, f_{k+m})$;
2. for any tuple $\overline{t} \subseteq \overline{ae}$, $\overline{t} \in R_i^{\mathcal{A}}$ iff $h(\overline{t}) \in R_i^{\mathcal{B}}$.

For a fixed $k \geq 0$, the Ehrenfeucht-Fraïssé game of length k is played for k rounds. We say that the duplicator *wins* the game if he has a strategy to win every round. As a special case, when $|\overline{a}| = |\overline{b}| = 0$, we also say that the duplicator *wins* the Ehrenfeucht-Fraïssé game of length k on $\mathcal{A}$ and $\mathcal{B}$.

Theorem 1 *(Ebbinghaus & Flum 1999) The duplicator wins the Ehrenfeucht-Fraïssé game of length k played on $\mathcal{A}$ and $\mathcal{B}$, iff $\mathcal{A} \equiv_k \mathcal{B}$.*

Then we can prove the following theorem to characterize the first-order definability for a given program.

Theorem 2 *Let Π be a program. Π is not first-order definable if and only if for every $k \geq 0$, there are two structures $\mathcal{A}^k$ and $\mathcal{B}^k$ of vocabulary $\tau(\Pi)$ such that*[1]*:*

1. *$\mathcal{A}^k \in \Im(\Pi, \mathcal{A}^k|\tau_{ext}(\Pi))$, $\mathcal{B}^k \notin \Im(\Pi, \mathcal{B}^k|\tau_{ext}(\Pi))$; and*
2. *the duplicator wins the Ehrenfeucht-Fraïssé game of length k on $\mathcal{A}^k$ and $\mathcal{B}^k$.*

The program of finding Hamiltonian cycles has been used as a benchmark to test various ASP solvers. As an application of Theorem 2, we will show that this program is not first-order definable.

Proposition 1 *The following finding Hamiltonian cycles program Π_{HC} is not first-order definable:*

$$
\begin{array}{l}
HC(x,y) \leftarrow E(x,y), \mathtt{not}\ OtherRoute(x,y),\\
OtherRoute(x,y) \leftarrow\\
\qquad E(x,y), E(x,z), HC(x,z), y \neq z,\\
OtherRoute(x,y) \leftarrow\\
\qquad E(x,y), E(z,y), HC(z,y), x \neq z,\\
Reached(y) \leftarrow E(x,y), HC(x,y),\\
\qquad Reached(x), \mathtt{not}\ InitialVertex(x),\\
Reached(y) \leftarrow\\
\qquad E(x,y), HC(x,y), InitialVertex(x),\\
\leftarrow \mathtt{not}\ Reached(x).
\end{array}
$$

Proof: (*Sketch*) For each $k \geq 0$, we consider two structures $\mathcal{A}^k$ and $\mathcal{B}^k$ of $\tau(\Pi)$, where

$$
\begin{array}{l}
\mathsf{Dom}(\mathcal{A}^k) = A^k = \{0, 1 \cdots, 2m-1\}, m \geq 2^{k+1},\\
E^{\mathcal{A}^k} = \{(i, i+1) \mid 0 \leq i < (2m-1)\} \cup \{(2m-1, 0)\},\\
InitialVertex^{\mathcal{A}^k} = \{0\}, \qquad HC^{\mathcal{A}^k} = E^{\mathcal{A}^k},\\
OtherRoute^{\mathcal{A}^k} = \emptyset,\\
Reached^{\mathcal{A}^k} = \{0, 1, \cdots, 2m-1\},\\
\mathsf{Dom}(\mathcal{B}^k) = \{0, 1, \cdots, 2m-1\},\\
E^{\mathcal{B}^k} = \{(i, i+1) \mid 0 \leq i < (m-1)\} \cup \{(m-1, 0)\} \cup\\
\qquad \{(j, j+1) \mid m \leq j < (2m-1)\} \cup\\
\qquad \{(2m-1, m)\},\\
InitialVertex^{\mathcal{B}^k} = \{0\}, \qquad HC^{\mathcal{B}^k} = E^{\mathcal{B}^k},\\
OtherRoute^{\mathcal{B}^k} = \emptyset,\\
Reached^{\mathcal{B}^k} = \{0, 1, \cdots, 2m-1\}.
\end{array}
$$

Note that if we only consider the extensional relations, $\mathcal{A}^k$ and $\mathcal{B}^k$ may be viewed as two different graphs with $\mathsf{Dom}(\mathcal{A}^k)$ and $\mathsf{Dom}(\mathcal{B}^k)$ being their vertices and $E^{\mathcal{A}^k}$ and $E^{\mathcal{B}^k}$ being their edges respectively. Furthermore, $(\mathsf{Dom}(\mathcal{A}^k), E^{\mathcal{A}^k})$ is a single cycle of length $2m$, and $(\mathsf{Dom}(\mathcal{B}^k), E^{\mathcal{B}^k})$ contains two separate cycles and each has a length m.

From the the interpretations of all intentional predicates in $\mathcal{A}^k$, it is easy to see that $\mathcal{A}^k$ is an answer set of Π_{HC}. On the other hand, $\mathcal{B}^k$ is not an answer set of Π_{HC} because $Reached^{\mathcal{B}^k} = \{0, 1, \cdots, 2m-1\}$, while it is observed that for each j ($j \geq m$), j is not reachable under

[1] It is important to note that this theorem is different from the general form of Ehrenfeucht-Fraïssé game theorem (Ebbinghaus & Flum 1999), where it is required that $\Im(\Pi, \mathcal{A}^k|\tau_{ext}(\Pi))$ and $\Im(\Pi, \mathcal{B}^k|\tau_{ext}(\Pi))$ must be the same class of structures. This is not the case here.

the given $E^{\mathcal{B}^k}$ and $InitialVertex^{\mathcal{B}^k}$. So we have $\mathcal{A}^k \in \Im(\Pi, \mathcal{A}^k|\tau_{ext}(\Pi_{HC}))$ and $\mathcal{B}^k \notin \Im(\Pi, \mathcal{B}^k|\tau_{ext}(\Pi_{HC}))$.

Now we consider the Ehrenfeucht-Fraïssé game of length k played on $\mathcal{A}^k$ and $\mathcal{B}^k$. Without loss of generality, we assume that the game starts with two special points played in each of the graph: $a_{-1} = 0, a_0 = (2m-1)$ from $\mathcal{A}^k$, and their responses $b_{-1} = 0, b_0 = (m-1)$ from $\mathcal{B}^k$ respectively. Intuitively, this means that the two endpoints of the cycle in $\mathcal{A}^k$ have responses of the two endpoints of one cycle in $\mathcal{B}^k$. Then during the game is played, we denote that a point a_i from $\mathcal{A}^k$ has its response b_i from $\mathcal{B}^k$, and *vice versa*. We also define the *distance* between two points in $\mathcal{A}^k$ or $\mathcal{B}^k$ to be the shortest path between them. Note that in $\mathcal{B}^k$, if one point is in one cycle component and the other is in another cycle component, the distance between these two points is infinity.

In order to prove that Π_{HC} is not first-order definable, according to Theorem 2, we only need to show that the duplicator has a winning strategy.

By induction, we can prove that the duplicator can play the game in such a way that ensures the following conditions after each round i[2]:

Condition 1. If $d(a_j, a_l) \leq 2^{k-i}$,
then $d(b_j, b_l) = d(a_j, a_l)$,
Condition 2. If $d(a_j, a_l) > 2^{k-i}$,
then $d(b_j, b_l) > 2^{k-i}$.

Finally, we further show that for each k, the duplicator wins the game of length k. From Theorem 1, that is, we need to prove $\mathcal{A}^k \equiv_k \mathcal{B}^k$. More specifically, we show that after k rounds, for any a_i, a_j from $\mathcal{A}^k$ and the corresponding b_i, b_j from $\mathcal{B}^k$, the following statements hold:

(1) $(a_i, a_j) \in E^{\mathcal{A}^k}$ iff $(b_i, b_j) \in E^{\mathcal{B}^k}$,
(2) $a_i \in InitialVertex^{\mathcal{A}^k}$ iff $b_i \in InitialVertex^{\mathcal{B}^k}$,
(3) $(a_i, a_j) \in HC^{\mathcal{A}^k}$ iff $(b_i, b_j) \in HC^{\mathcal{B}^k}$,
(4) $(a_i, a_j) \in OtherRoute^{\mathcal{A}^k}$ iff
$(b_i, b_j) \in OtherRoute^{\mathcal{B}^k}$, and
(5) $a_i \in Reached^{\mathcal{A}^k}$ iff $b_i \in Reached^{\mathcal{B}^k}$.

According to Conditions 1 and 2 we proved above, and the construction of $\mathcal{A}^k$ and $\mathcal{B}^k$, it can be verified that (1)-(5) hold. □

Sufficient Conditions for Proving ASP First-order Indefinability

From the proof of Proposition 1, it is observed that showing a program to be first-order indefinable is rather technical. In particular, during an Ehrenfeucht-Fraïssé game playing, the winning strategy for the duplicator highly relies on the structures we pick up for the proof. In this sense, the approach demonstrated in the proof of Proposition 1 would be hardly applied as a general approach to show indefinability for other programs.

[2]Note that only these two conditions will be sufficient to lead to our solution. Also, due to a space limit, we omit the detailed proof of these conditions.

On the other hand, existing results in finite model theory regarding the sufficient conditions to ensure winning strategies in Ehrenfeucht-Fraïssé games, for instance, those results developed in (Arora & Fagin 1997), are just too general to apply under our ASP setting.

Taking a closer look at the proof of Proposition 1, we observe that there seem to have two important factors to effectively apply the Ehrenfeucht-Fraïssé game technique: (1) both the given program's intentional and extensional relations have to be considered during the game; and (2) the embedded structural form (e.g. a cycle) of extensional relations also significantly affects the duplicator's winning strategy in the game. Based on these observations, we will develop useful sufficient conditions for proving a program's first-order indefinability which are easier to use in various situations.

Programs with the 0-1 property

Let $\mathcal{A} = (A, c_1^{\mathcal{A}}, \cdots, c_m^{\mathcal{A}}, R_1^{\mathcal{A}}, \cdots, R_n^{\mathcal{A}})$ be a structure. A relation $R_i^{\mathcal{A}}$ in $\mathcal{A}$ is called *0-relation* if $R_i^{\mathcal{A}} = \emptyset$, it is called *1-relation* if $R_i^{\mathcal{A}} = A^h$, were h is the arity of R_i. In general, a relation $R_i^{\mathcal{A}}$ in $\mathcal{A}$ is called *0-1 relation* if it is either a 0-relation or a 1-relation.

Definition 3 *(**The 0-1 property**) We say that program* Π *has the* 0-1 property*, if for each* $k \geq 1$*,* Π *has an answer set* $\mathcal{A}$*, where* $|\mathsf{Dom}(\mathcal{A})| \geq k$*, such that all intentional relations in* $\mathcal{A}$ *are 0-1 relations. In this case, we also call* $\mathcal{A}$ *a* 0-1 answer set *of* Π *and* Π *a* 0-1 program*.*

Example 1 We consider program $\Pi_{RChecking}$ which checks whether each vertex in a graph is reachable from the given initial vertex (vertices):

$Reachable(x) \leftarrow InitialVertex(x)$,
$Reachable(y) \leftarrow Reachable(x), E(x, y)$,
$\leftarrow$ not $Reachable(x)$.

We can see that for each $k \geq 0$, there exists an answer set of $\Pi_{RChecking}$, such that the intentional predicate $Reachable$'s interpretation in the answer set represents a 1-relation. Hence, $\Pi_{RChecking}$ has the 0-1 property. □

0-1 programs represent an important feature which will ensure the duplicator's winning strategy in an overall Ehrenfeucht-Fraïssé game based on certain local information. In particular, if a program has the 0-1 property, all we need to consider during an Ehrenfeucht-Fraïssé game playing is the underlying program's extensional relations in relevant structures/answer sets.

Theorem 3 *(**The 0-1 theorem**) Let* Π *be a 0-1 program.* Π *is not first-order definable if for each* $k \geq 0$*, there exists a structure* $\mathcal{B}$ *of* $\tau(\Pi)$*, such that* $\mathcal{B}$ *is not an answer set of* Π*, and* $\mathcal{A}|_{\tau_{ext}(\Pi)} \equiv_k \mathcal{B}|_{\tau_{ext}(\Pi)}$*, where* $\mathcal{A}$ *is a 0-1 answer set of* Π*, and for each* $P \in \tau_{int}(\Pi)$*,* $P^{\mathcal{B}} = P^{\mathcal{A}}$*.*

By Theorem 3, if a program has the 0-1 property, then when we prove the program's first-order indefinability, we may only apply the Ehrenfeucht-Fraïssé game over the restricted structures generated by extensional relations, e.g. $\mathcal{A}|_{\tau_{ext}(\Pi)}$ and $\mathcal{B}|_{\tau_{ext}(\Pi)}$, instead of the whole structures, which are usually simpler. This is because in general, extensional relations for a program can be arbitrary. Consequently, their corresponding structures are also allowed to

be flexible so that an Ehrenfeucht-Fraïssé game is easier to be proposed on such flexible structures. The following example shows an application of Theorem 3.

Example 2 (Example 1 continued). We show that $\Pi_{RChecking}$ is not first-order definable. In Example 1, we showed that $\Pi_{RChecking}$ satisfies the 0-1 property. From Theorem 3, all we need to do is that for each k, we can construct two structures $\mathcal{A}^k$ and $\mathcal{B}^k$ such that (1) $\mathcal{A}^k$ is an answer set while $\mathcal{B}^k$ is not; (2) $Reachable^{\mathcal{A}^k}$ and $Reachable^{\mathcal{B}^k}$ are the 1-relations in $\mathcal{A}^k$ and $\mathcal{B}^k$ respectively; and (3) prove $\mathcal{A}^k|_{\tau_{ext}(\Pi_{RChecking})} \equiv_k \mathcal{B}^k|_{\tau_{ext}(\Pi_{RChecking})}$, which can be showed using a similar method as described in the proof of Proposition 1. □

Programs with 0-1 unbounded cycles or paths

Theorem 3 can be effective in proving a 0-1 program Π's first-order indefinability if the proof of $\mathcal{A}^k|_{\tau_{ext}(\Pi)} \equiv_k \mathcal{B}^k|_{\tau_{ext}(\Pi)}$ is already clear through the Ehrenfeucht-Fraïssé game approach. Nevertheless, as has been revealed in finite model theory, directly using the Ehrenfeucht-Fraïssé game approach is technically challenging for general cases (Arora & Fagin 1997). Furthermore, in our first-order indefinability proofs for programs Π_{HC} and $\Pi_{RChecking}$, both programs happen to only have one binary extensional predicates, so that we can use graph representations to specify the game, which makes our proofs easier.

Although Theorems 2 and 3 do not rely on graph representations of structures, when a program involves more than one binary extensional predicates or extensional predicates with arity greater than 2, it does not seem to be obvious to use our method demonstrated in the proof of Proposition 2 to show a program's first-order indefinability.

In this subsection, we will develop another sufficient condition by which we can effectively prove a program's first-order indefinability under certain conditions.

To begin with, we first introduce a useful notion. Let $\mathcal{A}$ be a structure, the *Gaifman graph* of $\mathcal{A}$ (Ebbinghaus & Flum 1999) is an undirected graph $G(\mathcal{A}) = (A, Edge^{\mathcal{A}})$, where $\mathsf{Dom}(\mathcal{A}) = A$, and $Edge^{\mathcal{A}}$ is defined as follows:

$$Edge^{\mathcal{A}} = \{(a,b) \mid a \neq b \text{ and there are a relation } R^{\mathcal{A}} \text{ in } \mathcal{A} \text{ and } \overline{c} \text{ in } A \text{ such that } \overline{c} \in R^{\mathcal{A}} \text{ and } a \text{ and } b \text{ are among } \overline{c}\}.$$

We say that $\mathcal{A}$ has a *cycle* (or an *acyclic path*[3]) if $G(\mathcal{A})$ contains a connected component that is a cycle (or a path, resp.).

Definition 4 ***(Programs with 0-1 unbounded cycles or paths)*** *A program Π has* unbounded cycles *(or* paths*) if for every $k > 0$, there is a Π's answer set $\mathcal{A}$ such that $G(\mathcal{A}|_{\tau_{ext}(\Pi)})$ contains a cycle (path, resp.) with length greater than k. A program Π has 0-1* unbounded cycles *(or* paths*) if Π is a 0-1 program, and for every $k > 0$, there is a Π's 0-1 answer set $\mathcal{A}$ such that $G(\mathcal{A}|_{\tau_{ext}(\Pi)})$ contains a cycle (path, resp.) with length greater than k. In this case, $\mathcal{A}$ is called a 0-1* cyclic *(*linear*, resp.) answer set of Π.*

[3]We will simply call it a path.

Programs with 0-1 unbounded cycles or paths are of special interests in relation to first-order indefinability. The following theorem provides a new sufficient condition, which, as will be showed next, completely avoids the Ehrenfeucht-Fraïssé game.

Theorem 4 ***(The 0-1 unbounded cycles or paths theorem)*** *A program Π is not first-order definable if (1) Π has 0-1 unbounded cycles or paths, and (2) for each Π's 0-1 cyclic or linear answer set $\mathcal{A}$, $G(\mathcal{A}|_{\tau_{ext}(\Pi)})$ contains only* one *cycle or path, while all other connected components of $G(\mathcal{A}|_{\tau_{ext}(\Pi)})$ are neither cycles nor paths.*

Example 3 Consider program $\Pi_{TCovered}$ as follows:

r_1: $T(x,y) \leftarrow E(x,y), \texttt{not}\ E(x,x), \texttt{not}\ E(y,y)$,
r_2: $T(x,y) \leftarrow T(x,z), T(z,y)$,
r_3: $Covered(x) \leftarrow D(x,y)$,
r_4: $Covered(y) \leftarrow D(x,y)$,
r_5: $\leftarrow D(x,y), \texttt{not}\ E(x,y)$,
r_6: $\leftarrow \texttt{not}\ Covered(x)$.

Intuitively, program $\Pi_{TCovered}$ computes the transitive closure based on the subgraph of E without self-loops and verifies whether all vertices of the graph are covered by a given subset D of edges of the graph. □

Proposition 2 *Program $\Pi_{TCovered}$ in Example 3 is not first-order definable.*

Proof: We prove this result by using Theorem 4. For any given $k > 0$, we consider structure $\mathcal{A}^k$ as follows:

$\mathsf{Dom}(\mathcal{A}^k) = \{0, 1, \cdots, m\}$, where $m \geq k$,
$E^{\mathcal{A}^k} = \{(i, i+1) \mid 0 \leq i < m\} \cup \{(m, 0)\}$,
$D^{\mathcal{A}^k} = \{(j, j+1) \mid 0 \leq j < m\}$,
$T^{\mathcal{A}^k} = \{(i,j) \mid 0 \leq i, j \leq m\}$,
$Covered^{\mathcal{A}^k} = \{0, \cdots, m\}$.

It is easy to verify that $\mathcal{A}^k$ is a 0-1 answer set of $\Pi_{TCovered}$. In fact, for both intentional predicates T and $Covered$, they are interpreted as 1-relations in $\mathcal{A}^k$. So $\Pi_{TCovered}$ has the 0-1 property. Furthermore, $G(\mathcal{A}^k|_{\{E,D\}})$ is a cycle with length m. Since there is no bound on m, $\Pi_{TCovered}$ has 0-1 unbounded cycles.

It is also observed that for an arbitrary 0-1 cyclic answer set $\mathcal{B}$ of $\Pi_{TCovered}$, $G(\mathcal{B}^k|_{\{E,D\}})$ must be of the same form of $G(\mathcal{A}^k|_{\{E,D\}})$ as specified above. So both conditions (1) and (2) in Theorem 4 for $\Pi_{TCovered}$. This concludes that $\Pi_{TCovered}$ is not first-order definable. □

Proof of Theorem 4

In order to prove Theorem 4, we will need a result in finite model theory (Fagin, Stockmeyer, & Vardi 1995). We first present necessary notions and concepts. Consider a structure $\mathcal{A} = (A, c_1^{\mathcal{A}}, \cdots, c_m^{\mathcal{A}}, R_1^{\mathcal{A}}, \cdots, R_n^{\mathcal{A}})$. Let $G(\mathcal{A}) = (A, Edge^{\mathcal{A}})$ be the Gaifman graph of $\mathcal{A}$ and a an element of A. The *neighborhood* $N(a,d)$ of a of radius d is recursively defined as follows:

$N(a,1) = \{a, c_1^{\mathcal{A}}, \cdots, c_m^{\mathcal{A}}\}$,
$N(a,d+1) = N(a,d) \cup \{c \mid c \in A$, and there is $b \in N(a,d)$ such that $(b,c) \in Edge^{\mathcal{A}}\}$.

Intuitively, $N(a,d)$ may be viewed as a sphere forming from elements of A where each element in $N(a,d)$ has a *distance* from a not more than d. Then we define that the d*-type* of a is the isomorphism type of $\mathcal{A} \uparrow N(a,d)$. That is, if $\mathcal{B}$ is a structure of the same vocabulary of $\mathcal{A}$ and b is an element of $\mathsf{Dom}(\mathcal{B})$, then a and b have the same d-type iff $\mathcal{A} \uparrow N(a,d) \cong \mathcal{B} \uparrow N(b,d)$ under an isomorphism mapping a to b. $\mathcal{A}$ and $\mathcal{B}$ are d*-equivalent* if for every d-type ι, they have the same number of points with d-type ι.

Theorem 5 *(Fagin, Stockmeyer, & Vardi 1995) For every $k > 0$ and for every $d \geq 3^{k-1}$, if $\mathcal{A}$ and $\mathcal{B}$ are d-equivalent, then $\mathcal{A} \equiv_k \mathcal{B}$.*

Lemma 1 *If Π has unbounded cycles, then for each $k > 0$, there exist two structures $\mathcal{A}$ and $\mathcal{B}$ of $\tau(\Pi)$ such that (1) $\mathcal{A}$ is an answer set of Π and $G(\mathcal{A}|_{\tau_{ext}(\Pi)})$ contains a cycle, (2) $G(\mathcal{B}|_{\tau_{ext}(\Pi)})$ contains two disjoint cycles, and (3) for each $d > 0$, $\mathcal{A}|_{\tau_{ext}(\Pi)}$ and $\mathcal{B}|_{\tau_{ext}(\Pi)}$ are d-equivalent.*

Lemma 2 *If Π has unbounded paths, then for each $k > 0$, there exist two structures $\mathcal{A}$ and $\mathcal{B}$ of $\tau(\Pi)$ such that (1) $\mathcal{A}$ is an answer set of Π and $G(\mathcal{A}|_{\tau_{ext}(\Pi)})$ contains a path, (2) $G(\mathcal{B}|_{\tau_{ext}(\Pi)})$ contains disjoint one cycle and one path, and (3) for each $d > 0$, $\mathcal{A}|_{\tau_{ext}(\Pi)}$ and $\mathcal{B}|_{\tau_{ext}(\Pi)}$ are d-equivalent.*

Proof of Theorem 4:
Since Π is a 0-1 program and has 0-1 unbounded cycles or paths, from Lemmas 1 and 2, we know that for any $k > 0$, Π has a 0-1 cyclic or linear answer set $\mathcal{A}$, and we can always find a structure $\mathcal{B}$ of $\tau(\Pi)$ such that $\mathcal{A}|_{\tau_{ext}(\Pi)}$ and $\mathcal{B}|_{\tau_{ext}(\Pi)}$ are d-equivalent for each $d > 0$, where whenever $G(\mathcal{A}|_{\tau_{ext}(\Pi)})$ contains one cycle (path), $G(\mathcal{B}|_{\tau_{ext}(\Pi)})$ contains two cycles (one cycle and one path, resp.). Since this result holds for any $k > 0$ and $d > 0$, from Theorem 5, by setting $d \geq 3^{k-1}$, we then have $\mathcal{A}|_{\tau_{ext}(\Pi)} \equiv_k \mathcal{B}|_{\tau_{ext}(\Pi)}$.

Now by setting every intentional relation of $\mathcal{B}$ to be either 0-relation or 1-relation accordingly as in $\mathcal{A}$, it is concluded that $\mathcal{B}$ cannot be an answer set of Π due to condition (2) of Theorem 4. So by Theorem 3, Π is not first-order definable. □

Related Work

In Datalog, it has been shown that on arbitrary structures, a datalog program is bounded iff the corresponding datalog queries are first-order definable iff the datalog program is equivalent to a recursion-free datalog program. On finite structures, this is also true for pure datalog programs but not for arbitrary datalog programs (Ajtai & Gurevich 1994; Cosmadakis 1989). These results, however, do not provide many hints about how to prove a datalog query's first-order (in)definability.

In finite model theory, Ehrenfeucht-Fraïssé game approach is the primary tool for proving first-order indefinability result (Ebbinghaus & Flum 1999). However, as it is well recognized (Fagin 1997), using this approach for specific cases is technically challenging. One way to deal with such difficulty is to develop stronger sufficient conditions to ensure the winning strategy for the duplicator during an Ehrenfeucht-Fraïssé game. Although the existing results in finite model theory, for instance, those summarized in (Fagin 1997), are just too general to apply to our problems under ASP setting, this idea indeed motivated our work presented in this paper.

The first result of extending Ehrenfeucht-Fraïssé game approach to Datalog was due to Cosmadakis' work (Cosmadakis 1989). Our Theorem 2 may be viewed as an analogy of Theorem 2.6 in (Cosmadakis 1989) for ASP. Note that both results have looser conditions for the classes of structures than the original Ehrenfeucht-Fraïssé game theorem (Ebbinghaus & Flum 1999). Since these two results only provide the corresponding Ehrenfeucht-Fraïssé game-theoretic characterizations on the frist-order indefinability for ASP and Datalog respectively, as in finite model theory, they are quite hard to use.

Conclusions

In this paper, we provided an Ehrenfeucht-Fraïssé game-theoretic characterization on the first-order indefinability of answer set programs on finite structures. The two sufficient conditions proposed in this paper can be used as powerful tools in proving an answer set program's first-order indefinability. In fact, we have further generalized the two sufficient conditions (Theorems 3 and 4) and discovered that most commonly known first-order indefinable programs have been covered by our results. Due to a space limit, we refer this part to our full paper.

Acknowledgement: The first author was partially supported by a grant of National Science Foundation of China (NSFC60705095). This research was supported in part by an Australian Research Council Discovery grant (DP0988396).

References

Ajtai, M., and Gurevich, Y. 1994. Datalog vs. first order logic. *J. of Computer and System Sciences* 49:562–588.

Arora, S., and Fagin, R. 1997. On winning strategies in Ehrenfeucht-Fraïssé games. *Theoretical Computer Science* 174:97–121.

Baral, C. 2003. *Knowledge Representsation, Reasoning, and Declarative Problem Solving*. MIT Press.

Cosmadakis, S. 1989. On the first-order experssibility of recursive queries. In *Proceedings of the 8th ACM SIGACT-SIGMOD-SIGART Symposium on PODS*, 311–323.

Dantsin, E., and et al. 2001. Complexity and expressive power of logic programming. *ACM Computing Surverys* 33:374–425.

Ebbinghaus, H., and Flum, J. 1999. *Finite Model Theory*. 2nd edition, Springer.

Fagin, R.; Stockmeyer, L.; and Vardi, M. 1995. On monadic NP vs. monadic co-NP. *Information and Computation* 120:78–92.

Fagin, R. 1997. Easier ways to win logical games. *Descriptive Complexity and Finite Models* 1–32.

Ferraris, P.; Lee, J.; and Lifschitz, V. 2010. Stabel models and circumscription. *Artificial Intelligence*.

Lin, F., and Zhou, Y. 2007. From answer set logic programming to circumscription via logic of gk. In *Proceedings of IJCAI-2007*, 441–661.