Model Completeness, Covers and Superposition

Diego Calvanese[1], Silvio Ghilardi[2], Alessandro Gianola[1](✉), Marco Montali[1], and Andrey Rivkin[1]

[1] Faculty of Computer Science, Free University of Bozen-Bolzano, Bolzano, Italy
{calvanese,gianola,montali,rivkin}@inf.unibz.it
[2] Dipartimento di Matematica, Università degli Studi di Milano, Milan, Italy
silvio.ghilardi@unimi.it

Abstract. In ESOP 2008, Gulwani and Musuvathi introduced a notion of cover and exploited it to handle infinite-state model checking problems. Motivated by applications to the verification of data-aware processes, we show how covers are strictly related to model completions, a well-known topic in model theory. We also investigate the computation of covers within the Superposition Calculus, by adopting a constrained version of the calculus, equipped with appropriate settings and reduction strategies.

1 Introduction

Declarative approaches to infinite state model checking [40] need to manipulate logical formulae in order to represent sets of reachable states. To prevent divergence, various abstraction strategies have been adopted, ranging from interpolation-based [33] to sophisticated search via counterexample elimination [26]. Precise computations of the set of reachable states require some form of quantifier elimination and hence are subject to two problems, namely that quantifier elimination might not be available at all and that, when available, it is computationally very expensive.

To cope with the first problem, [25] introduced the notion of a *cover* and proved that covers exist for equality with uninterpreted symbols (EUF) and its combination with linear arithmetic; also, it was shown that covers can be used instead of quantifier elimination and yield a precise computation of reachable states. Concerning the second problem, in [25] it was observed (as a side remark) that computing the cover of a conjunction of literals becomes tractable when only free unary function symbols occur in the signature. It can be shown (see [10]) that the same observation applies when also free relational symbols occur.

In [11,12] we propose a new formalism for representing *read-only database schemata* towards the verification of integrated models of processes and data [9], in particular so-called *artifact systems* [7,15,31,43]; this formalism (briefly recalled in Sect. 4.1 below) uses precisely signatures comprising unary function

P. Fontaine (Ed.): CADE 2019, LNAI 11716, pp. 142–160, 2019.
https://doi.org/10.1007/978-3-030-29436-6_9

symbols and free n-ary relations. In [11,12] we apply model completeness techniques for verifying transition systems based on read-only databases, in a framework where such systems employ both individual and higher order variables.

In this paper we show (see Sect. 3 below) that covers are strictly related to *model completions* and to *uniform interpolation* [39], thus building a bridge between different research areas. In particular, we prove that computing covers for a theory is *equivalent* to eliminating quantifiers in its model completion. Model completeness has other well-known applications in computer science. It has been applied: *(i)* to reveal interesting connections between temporal logic and monadic second order logic [22,23]; *(ii)* in automated reasoning to design complete algorithms for constraint satisfiability in combined theories over non disjoint signatures [1,17,20,34–36] and theory extensions [41,42]; *(iii)* to obtain combined interpolation for modal logics and software verification theories [18,19].

In the last part of the paper (Sect. 5 below), we prove that covers for EUF can be computed through a constrained version of the *Superposition Calculus* [38] equipped with appropriate settings and reduction strategies; the related completeness proof requires a careful analysis of the constrained literals generated during the saturation process. Not all proofs could be included here: for the missing ones, we refer to the online available extended version [10] (the proofs of our results from Sect. 5 are however reported in full detail).

2 Preliminaries

We adopt the usual first-order syntactic notions of signature, term, atom, (ground) formula, and so on; our signatures are multi-sorted and include equality for every sort. Hence variables are sorted as well. For simplicity, some basic definitions will be supplied for single-sorted languages only (the adaptation to multi-sorted languages is straightforward). We compactly represent a tuple $\langle x_1, \ldots, x_n \rangle$ of variables as $\underline{x}$. The notation $t(\underline{x}), \phi(\underline{x})$ means that the term t, the formula ϕ has free variables included in the tuple $\underline{x}$. We assume that a function arity can be deduced from the context. Whenever we build terms and formulae, we always assume that they are well-typed, i.e., that the sorts of variables, constants, and function sources/targets match. A formula is said to be *universal* (resp., *existential*) if it has the form $\forall \underline{x}(\phi(\underline{x}))$ (resp., $\exists \underline{x}(\phi(\underline{x}))$), where ϕ is a quantifier-free formula. Formulae with no free variables are called *sentences*. From the semantic side, we use the standard notion of Σ-structure $\mathcal{M}$ and of truth of a formula in a Σ-structure under a free variables assignment. The *support* $|\mathcal{M}|$ of $\mathcal{M}$ is the disjoint union of the interpretations of the sorts in Σ. The interpretation of a (sort, function, predicate) symbol σ in $\mathcal{M}$ is denoted $\sigma^{\mathcal{M}}$.

A *Σ-theory* T is a set of Σ-sentences; a *model* of T is a Σ-structure $\mathcal{M}$ where all sentences in T are true. We use the standard notation $T \models \phi$ to say that ϕ is true in all models of T for every assignment to the variables occurring free in ϕ. We say that ϕ is *T-satisfiable* iff there is a model $\mathcal{M}$ of T and an assignment to the variables occurring free in ϕ making ϕ true in $\mathcal{M}$.

We now focus on the constraint satisfiability problem and quantifier elimination for a theory T. A Σ-formula ϕ is a *Σ-constraint* (or just a constraint) iff it is a conjunction of literals. The *constraint satisfiability problem* for T is the following: we are given a constraint (equivalently, a quantifier-free formula) $\phi(\underline{x})$ and we are asked whether there exist a model $\mathcal{M}$ of T and an assignment $\mathcal{I}$ to the free variables $\underline{x}$ such that $\mathcal{M}, \mathcal{I} \models \phi(\underline{x})$. A theory T has *quantifier elimination* iff for every formula $\phi(\underline{x})$ in the signature of T there is a quantifier-free formula $\phi'(\underline{x})$ such that $T \models \phi(\underline{x}) \leftrightarrow \phi'(\underline{x})$. Since we are in a computational logic context, when we speak of quantifier elimination, we assume that it is effective, namely that it comes with an algorithm for computing ϕ' out of ϕ. It is well-known that quantifier elimination holds in case we can eliminate quantifiers from *primitive* formulae, i.e., formulae of the kind $\exists \underline{y}\, \phi(\underline{x}, \underline{y})$, with ϕ a constraint.

We recall also some basic notions from logic and model theory. Let Σ be a first-order signature. The signature obtained from Σ by adding to it a set $\underline{a}$ of new constants (i.e., 0-ary function symbols) is denoted by $\Sigma^{\underline{a}}$. Analogously, given a Σ-structure $\mathcal{M}$, the signature Σ can be expanded to a new signature $\Sigma^{|\mathcal{M}|} := \Sigma \cup \{\bar{a} \mid a \in |\mathcal{M}|\}$ by adding a set of new constants $\bar{a}$ (the *name* for a), one for each element a in $\mathcal{M}$, with the convention that two distinct elements are denoted by different "name" constants. $\mathcal{M}$ can be expanded to a $\Sigma^{|\mathcal{M}|}$-structure $\overline{\mathcal{M}} := (\mathcal{M}, a)_{a \in |\mathcal{M}|}$ just interpreting the additional constants over the corresponding elements. From now on, when the meaning is clear from the context, we will freely use the notation $\mathcal{M}$ and $\overline{\mathcal{M}}$ interchangeably: in particular, given a Σ-structure $\mathcal{M}$ and a Σ-formula $\phi(\underline{x})$ with free variables that are all in $\underline{x}$, we will write, by abuse of notation, $\mathcal{M} \models \phi(\underline{a})$ instead of $\overline{\mathcal{M}} \models \phi(\underline{\bar{a}})$.

A *Σ-homomorphism* (or, simply, a homomorphism) between two Σ-structures $\mathcal{M}$ and $\mathcal{N}$ is a map $\mu : |\mathcal{M}| \longrightarrow |\mathcal{N}|$ among the support sets $|\mathcal{M}|$ of $\mathcal{M}$ and $|\mathcal{N}|$ of $\mathcal{N}$ satisfying the condition ($\mathcal{M} \models \varphi \quad \Rightarrow \quad \mathcal{N} \models \varphi$) for all $\Sigma^{|\mathcal{M}|}$-atoms φ ($\mathcal{M}$ is regarded as a $\Sigma^{|\mathcal{M}|}$-structure, by interpreting each additional constant $a \in |\mathcal{M}|$ into itself and $\mathcal{N}$ is regarded as a $\Sigma^{|\mathcal{M}|}$-structure by interpreting each additional constant $a \in |\mathcal{M}|$ into $\mu(a)$). In case the last condition holds for all $\Sigma^{|\mathcal{M}|}$-literals, the homomorphism μ is said to be an *embedding* and if it holds for all first order formulae, the embedding μ is said to be *elementary.* If $\mu : \mathcal{M} \longrightarrow \mathcal{N}$ is an embedding which is just the identity inclusion $|\mathcal{M}| \subseteq |\mathcal{N}|$, we say that $\mathcal{M}$ is a *substructure* of $\mathcal{N}$ or that $\mathcal{N}$ is an *extension* of $\mathcal{M}$.

Let $\mathcal{M}$ be a Σ-structure. The *diagram* of $\mathcal{M}$, written $\Delta_{\Sigma}(\mathcal{M})$ (or just $\Delta(\mathcal{M})$), is the set of ground $\Sigma^{|\mathcal{M}|}$-literals that are true in $\mathcal{M}$. An easy but important result, called *Robinson Diagram Lemma* [13], says that, given any Σ-structure $\mathcal{N}$, the embeddings $\mu : \mathcal{M} \longrightarrow \mathcal{N}$ are in bijective correspondence with expansions of $\mathcal{N}$ to $\Sigma^{|\mathcal{M}|}$-structures which are models of $\Delta_{\Sigma}(\mathcal{M})$. The expansions and the embeddings are related in the obvious way: $\bar{a}$ is interpreted as $\mu(a)$.

3 Covers, Uniform Interpolation and Model Completions

We report the notion of *cover* taken from [25]. Fix a theory T and an existential formula $\exists \underline{e}\, \phi(\underline{e}, \underline{y})$; call a *residue* of $\exists \underline{e}\, \phi(\underline{e}, \underline{y})$ any quantifier-free formula belonging to the set of quantifier-free formulae $Res(\exists \underline{e}\, \phi) = \{\theta(\underline{y}, \underline{z}) \mid T \models \phi(\underline{e}, \underline{y}) \to \theta(\underline{y}, \underline{z})\}$. A quantifier-free formula $\psi(\underline{y})$ is said to be a *T-cover* (or, simply, a *cover*) of $\exists \underline{e}\, \phi(\underline{e}, \underline{y})$ iff $\psi(\underline{y}) \in \mathit{Res}(\exists \underline{e}\, \phi)$ and $\psi(\underline{y})$ implies (modulo T) all the other formulae in $\mathit{Res}(\exists \underline{e}\, \phi)$. The following Lemma (to be widely used throughout the paper) supplies a semantic counterpart to the notion of a cover:

Lemma 1. *A formula $\psi(\underline{y})$ is a T-cover of $\exists \underline{e}\, \phi(\underline{e}, \underline{y})$ iff it satisfies the following two conditions: (i) $T \models \forall \underline{y}\, (\exists \underline{e}\, \phi(\underline{e}, \underline{y}) \to \psi(\underline{y}))$; (ii) for every model $\mathcal{M}$ of T, for every tuple of elements $\underline{a}$ from the support of $\mathcal{M}$ such that $\mathcal{M} \models \psi(\underline{a})$ it is possible to find another model $\mathcal{N}$ of T such that $\mathcal{M}$ embeds into $\mathcal{N}$ and $\mathcal{N} \models \exists \underline{e}\, \phi(\underline{e}, \underline{a})$.* $\lhd$

Proof. Suppose that $\psi(\underline{y})$ satisfies conditions (i) and (ii) above. Condition (i) says that $\psi(\underline{y}) \in \mathit{Res}(\exists \underline{e}\, \phi)$, so ψ is a residue. In order to show that ψ is also a cover, we have to prove that $T \models \forall \underline{y}, \underline{z}(\psi(\underline{y}) \to \theta(\underline{y}, \underline{z}))$, for every $\theta(\underline{y}, \underline{z})$ that is a residue for $\exists \underline{e}\, \phi(\underline{e}, \underline{y})$. Given a model $\mathcal{M}$ of T, take a pair of tuples $\underline{a}, \underline{b}$ of elements from $|\mathcal{M}|$ and suppose that $\mathcal{M} \models \psi(\underline{a})$. By condition (ii), there is a model $\mathcal{N}$ of T such that $\mathcal{M}$ embeds into $\mathcal{N}$ and $\mathcal{N} \models \exists \underline{e} \phi(\underline{e}, \underline{a})$. Using the definition of $\mathit{Res}(\exists \underline{e}\, \phi)$, we have $\mathcal{N} \models \theta(\underline{a}, \underline{b})$, since $\theta(\underline{y}, \underline{z}) \in \mathit{Res}(\exists \underline{x}\, \phi)$. Since $\mathcal{M}$ is a substructure of $\mathcal{N}$ and θ is quantifier-free, $\mathcal{M} \models \theta(\underline{a}, \underline{b})$ as well, as required.

Suppose that $\psi(\underline{y})$ is a cover. The definition of residue implies condition (i). To show condition (ii) we have to prove that, given a model $\mathcal{M}$ of T, for every tuple $\underline{a}$ of elements from $|\mathcal{M}|$, if $\mathcal{M} \models \psi(\underline{a})$, then there exists a model $\mathcal{N}$ of T such that $\mathcal{M}$ embeds into $\mathcal{N}$ and $\mathcal{N} \models \exists \underline{x} \phi(\underline{x}, \underline{a})$. By reduction to absurdity, suppose that this is not the case: this is equivalent (by using Robinson Diagram Lemma) to the fact that $\Delta(\mathcal{M}) \cup \{\phi(\underline{e}, \underline{a})\}$ is a T-inconsistent $\Sigma^{|\mathcal{M}| \cup \{\underline{e}\}}$-theory. By compactness, there is a finite number of literals $\ell_1(\underline{a}, \underline{b}), ..., \ell_m(\underline{a}, \underline{b})$ (for some tuple $\underline{b}$ of elements from $|\mathcal{M}|$) such that $\mathcal{M} \models \ell_i$ (for all $i = 1, \dots, m$) and $T \models \phi(\underline{e}, \underline{a}) \to \neg(\ell_1(\underline{a}, \underline{b}) \wedge \cdots \wedge \ell_m(\underline{a}, \underline{b}))$, which means that $T \models \phi(\underline{e}, \underline{y}) \to (\neg \ell_1(\underline{y}, \underline{z}) \vee \cdots \vee \neg \ell_m(\underline{y}, \underline{z}))$, i.e. that $T \models \exists \underline{e}\, \phi(\underline{e}, \underline{y}) \to (\neg \ell_1(\underline{y}, \underline{z}) \vee \cdots \vee \neg \ell_m(\underline{y}, \underline{z}))$. By definition of residue, clearly $(\neg \ell_1(\underline{y}, \underline{z}) \vee \cdots \vee \neg \ell_m(\underline{y}, \underline{z})) \in \mathit{Res}(\exists \underline{x}\, \phi)$; then, since $\psi(\underline{y})$ is a cover, $T \models \psi(\underline{y}) \to (\neg \ell_1(\underline{y}, \underline{z}) \vee \cdots \vee \neg \ell_m(\underline{y}, \underline{z}))$, which implies that $\mathcal{M} \models \neg \ell_j(\underline{a}, \underline{b})$ for some $j = 1, \dots, m$, which is a contradiction. Thus, $\psi(\underline{y})$ satisfies conditions (ii) too. $\dashv$

We say that a theory T has *uniform quantifier-free interpolation* iff every existential formula $\exists \underline{e}\, \phi(\underline{e}, \underline{y})$ (equivalently, every primitive formula $\exists \underline{e}\, \phi(\underline{e}, \underline{y})$) has a T-cover.

It is clear that if T has uniform quantifier-free interpolation, then it has ordinary quantifier-free interpolation [8], in the sense that if we have $T \models \phi(\underline{e}, \underline{y}) \to \phi'(\underline{y}, \underline{z})$ (for quantifier-free formulae ϕ, ϕ'), then there is a quantifier-free formula $\theta(\underline{y})$ such that $T \models \phi(\underline{e}, \underline{y}) \to \theta(\underline{y})$ and $T \models \theta(\underline{y}) \to \phi'(\underline{y}, \underline{z})$. In fact, if T has uniform quantifier-free interpolation, then the interpolant θ is independent on ϕ' (the same $\theta(\underline{y})$ can be used as interpolant for all entailments $T \models \phi(\underline{e}, \underline{y}) \to \phi'(\underline{y}, \underline{z})$, varying ϕ').

We say that a *universal* theory T has a *model completion* iff there is a stronger theory $T^* \supseteq T$ (still within the same signature Σ of T) such that (i) every Σ-constraint that is satisfiable in a model of T is satisfiable in a model of T^*; (ii) T^* eliminates quantifiers. Other equivalent definitions are possible [13]: for instance, (i) is equivalent to the fact that T and T^* prove the same quantifier-free formulae or again to the fact that every model of T can be embedded into a model of T^*. We recall that the model completion, if it exists, is unique and that its existence implies the amalgamation property for T [13]. The relationship between uniform interpolation in a propositional logic and model completion of the equational theory of the variety algebraizing it was extensively studied in [24]. In the context of first order theories, we prove an even more direct connection:

Theorem 1. *Suppose that T is a universal theory. Then T has a model completion T^* iff T has uniform quantifier-free interpolation. If this happens, T^* is axiomatized by the infinitely many sentences $\forall \underline{y}\,(\psi(\underline{y}) \to \exists \underline{e}\,\phi(\underline{e}, \underline{y}))$, where $\exists \underline{e}\,\phi(\underline{e}, \underline{y})$ is a primitive formula and ψ is a cover of it.* $\triangleleft$

The proof (via Lemma 1, by iterating a chain construction) is in [10].

4 Model-Checking Applications

In this section we supply old and new motivations for investigating covers and model completions in view of model-checking applications. We first report the considerations from [11,12,25] on symbolic model-checking via model completions (or, equivalently, via covers) in the basic case where system variables are represented as individual variables (for more advanced applications where system variables are both individual and higher order variables, see [11,12]). Similar ideas ('use quantifier elimination in the model completion even if T does not allow quantifier elimination') were used in [41] for interpolation and symbol elimination.

Definition 1. *A (quantifier-free)* transition system *is a tuple*

$$\mathcal{S} \;=\; \langle \Sigma, T, \underline{x}, \iota(\underline{x}), \tau(\underline{x}, \underline{x}')\rangle$$

where: (i) Σ is a signature and T is a Σ-theory; (ii) $\underline{x} = x_1, \ldots, x_n$ are individual variables; (iii) $\iota(\underline{x})$ is a quantifier-free formula; (iv) $\tau(\underline{x}, \underline{x}')$ is a quantifier-free formula (here the $\underline{x}'$ are renamed copies of the $\underline{x}$). $\triangleleft$

A *safety* formula for a transition system $\mathcal{S}$ is a further quantifier-free formula $\upsilon(\underline{x})$ describing undesired states of $\mathcal{S}$. We say that $\mathcal{S}$ is *safe with respect to υ* if the system has no finite run leading from ι to υ, i.e. (formally) if there are no model $\mathcal{M}$ of T and no $k \geq 0$ such that the formula

$$\iota(\underline{x}^0) \land \tau(\underline{x}^0, \underline{x}^1) \land \cdots \land \tau(\underline{x}^{k-1}, \underline{x}^k) \land \upsilon(\underline{x}^k) \tag{1}$$

is satisfiable in $\mathcal{M}$ (here $\underline{x}^i$'s are renamed copies of $\underline{x}$). The *safety problem* for $\mathcal{S}$ is the following: *given υ, decide whether $\mathcal{S}$ is safe with respect to υ.*

Suppose now that the theory T mentioned in Definition 1(i) is universal, has decidable constraint satisfiability problem and admits a model completion T^*. Algorithm 1 describes the *backward reachability algorithm* for handling the safety problem for $\mathcal{S}$ (the dual algorithm working via forward search is described in equivalent terms in [25]). An integral part of the algorithm is to compute preimages. For that purpose, for any $\phi_1(\underline{x}, \underline{x}')$ and $\phi_2(\underline{x})$, we define $Pre(\phi_1, \phi_2)$ to be the formula $\exists \underline{x}'(\phi_1(\underline{x}, \underline{x}') \land \phi_2(\underline{x}'))$. The *preimage* of the set of states described by a state formula $\phi(\underline{x})$ is the set of states described by $Pre(\tau, \phi)$. The subprocedure $\mathsf{QE}(T^*, \phi)$ in Line 6 applies the quantifier elimination algorithm of T^* to the existential formula ϕ. Algorithm 1 computes iterated preimages of υ and applies to them quantifier elimination, until a fixpoint is reached or until a set intersecting the initial states (i.e., satisfying ι) is found. *Inclusion* (Line 2) and *disjointness* (Line 3) tests produce proof obligations that can be discharged thanks to the fact that T has decidable constraint satisfiability problem.

Algorithm 1: Backward reachability algorithm

```
Function BReach($\upsilon$)
1   $\phi \longleftarrow \upsilon$; $B \longleftarrow \bot$;
2   while $\phi \land \neg B$ is $T$-satisfiable do
3       if $\iota \land \phi$ is $T$-satisfiable. then
            return unsafe
4       $B \longleftarrow \phi \lor B$;
5       $\phi \longleftarrow Pre(\tau, \phi)$;
6       $\phi \longleftarrow \mathsf{QE}(T^*, \phi)$;
    return (safe, $B$);
```

The proof of Proposition 1 consists just in the observation that, thanks to quantifier elimination in $T^\star$, (1) is a quantifier-free formula and that a quantifier-free formula is satisfiable in a model of T iff so is it in a model of T^*:

Proposition 1. *Suppose that the universal Σ-theory T has decidable constraint satisfiability problem and admits a model completion T^*. For every transition system $\mathcal{S} = \langle \Sigma, T, \underline{x}, \iota, \tau \rangle$, the backward search algorithm is effective and partially correct for solving safety problems for $\mathcal{S}$.*[1] $\lhd$

Despite its simplicity, Proposition 1 is a crucial fact. Notice that it implies decidability of the safety problems in some interesting cases: this happens, for instance, when in T there are only finitely many quantifier-free formulae in which $\underline{x}$ occur, as in case T has a purely relational signature or, more generally, T is *locally finite*[2]. Since a theory is universal iff it is closed under substructures [13] and since a universal locally finite theory has a model completion iff it has the amalgamation property [44], it follows that Proposition 1 can be used to cover the decidability result stated in Theorem 5 of [7] (once restricted to transition systems over a first-order definable class of Σ-structures).

[1] *Partial correctness* means that, when the algorithm terminates, it gives a correct answer. *Effectiveness* means that all subprocedures in the algorithm can be effectively executed.

[2] We say that T is locally finite iff for every finite tuple of variables $\underline{x}$ there are only finitely many non T-equivalent atoms $A(\underline{x})$ involving only the variables $\underline{x}$.

4.1 Database Schemata

In this subsection, we provide a new application for the above explained model-checking techniques [11,12]. The application relates to the verification of integrated models of business processes and data [9], referred to as artifact systems [43], where the behavior of the process is influenced by data stored in a relational database (DB) with constraints. The data contained therein are read-only: they can be queried by the process and stored in a working memory, which in the context of this paper is constituted by a set of system variables. In this context, safety amounts to checking whether the system never reaches an undesired property, irrespectively of what is contained in the read-only DB.

We define next the two key notions of (read-only) DB schema and instance, by relying on an algebraic, functional characterization.

Definition 2. *A* DB schema *is a pair $\langle \Sigma, T\rangle$, where: (i) Σ is a* DB signature*, that is, a finite multi-sorted signature whose function symbols are all unary; (ii) T is a* DB theory*, that is, a set of universal Σ-sentences.* ◁

We now focus on extensional data conforming to a given DB schema.

Definition 3. *A* DB instance *of DB schema $\langle \Sigma, T\rangle$ is a Σ-structure $\mathcal{M}$ such that $\mathcal{M}$ is a model of T.*[3] ◁

One might be surprised by the fact that signatures in our DB schemata contain unary function symbols, beside relational symbols. As shown in [11,12], the algebraic, functional characterization of DB schema and instance can be actually reinterpreted in the classical, relational model so as to reconstruct the requirements posed in [31]. Definition 2 naturally corresponds to the definition of relational database schema equipped with single-attribute *primary keys* and *foreign keys*. To see this connection, we adopt the *named perspective*, where each relation schema is defined by a signature containing a *relation name* and a set of *typed attribute names*. Let $\langle \Sigma, T\rangle$ be a DB schema. Each sort S from Σ corresponds to a dedicated relation R_S with the following attributes: (i) one identifier attribute id_S with type S; (ii) one dedicated attribute a_f with type S' for every function symbol f from Σ of the form $f : S \longrightarrow S'$.

The fact that R_S is constructed starting from functions in Σ naturally induces corresponding functional dependencies within R_S, and inclusion dependencies from R_S to other relation schemas. In particular, for each non-id attribute a_f of R_S, we get a functional dependency from id_S to a_f. Altogether, such dependencies witness that id_S is the *primary key* of R_S. In addition, for each non-id attribute a_f of R_S whose corresponding function symbol f has id sort S' as image, we get an inclusion dependency from a_f to the id attribute $id_{S'}$ of $R_{S'}$. This captures that a_f is a *foreign key* referencing $R_{S'}$.

[3] One may restrict to models interpreting sorts as *finite* sets, as customary in database theory. Since the theories we are dealing with usually have finite model property for constraint satisfiability, assuming such restriction turns out to be irrelevant, as far as safety problems are concerned (see [11,12] for an accurate discussion).

Given a DB instance $\mathcal{M}$ of $\langle \Sigma, T\rangle$, its corresponding relational instance $\mathcal{R}[\mathcal{M}]$ is the minimal set satisfying the following property: for every id sort S from Σ, let $f_1, \dots, f_n$ be all functions in Σ with domain S; then, for every identifier $\mathtt{o} \in S^{\mathcal{M}}$, $\mathcal{R}[\mathcal{M}]$ contains a *labeled fact* of the form $R_S(id_S : \mathtt{o}^{\mathcal{M}}, a_{f_1} : f_1^{\mathcal{M}}(\mathtt{o}), \dots, a_{f_n} : f_n^{\mathcal{M}}(\mathtt{o}))$. In addition, $\mathcal{R}[\mathcal{M}]$ contains the tuples from $r^{\mathcal{M}}$, for every relational symbol r from Σ (these relational symbols represent plain relations, i.e. those not possessing a key).

We close our discussion by focusing on DB theories. Notice that EUF suffices to handle the sophisticated setting of database-driven systems from [12] (e.g., key dependencies). The role of a non-empty DB theory is to encode background axioms to express additional constraints. We illustrate a typical background axiom, required to handle the possible presence of *undefined identifiers/values* in the different sorts. This, in turn, is essential to capture artifact systems whose working memory is initially undefined, in the style of [16,31]. To accommodate this, we add to every sort S of Σ a constant $\mathtt{undef}_S$ (written by abuse of notation just $\mathtt{undef}$ from now on), used to specify an undefined value. Then, for each function symbol f of Σ, we can impose additional constraints involving $\mathtt{undef}$, for example by adding the following axioms to the DB theory:

$$\forall x \; (x = \mathtt{undef} \leftrightarrow f(x) = \mathtt{undef}) \tag{2}$$

This axiom states that the application of f to the undefined value produces an undefined value, and it is the only situation for which f is undefined. A slightly different approach may handle *many* undefined values for each sort; the reader is referred to [11,12] for examples of concrete database instances formalized in our framework. We just point out that in most cases the kind of axioms that we need for our DB theories T are just *one-variable universal axioms* (like Axioms 2), so that they fit the hypotheses of Proposition 2 below.

We are interested in applying the algorithm of Proposition 1 to what we call *simple artifact systems*, i.e. transition systems $\mathcal{S} = \langle \Sigma, T, \underline{x}, \iota(\underline{x}), \tau(\underline{x}, \underline{x}')\rangle$, where $\langle \Sigma, T\rangle$ is a DB schema in the sense of Definition 2. To this aim, it is sufficient to identify a suitable class of DB theories having a model completion and whose constraint satisfiability problem is decidable. A first result in this sense is given below. We associate to a DB signature Σ the edge-labeled graph $G(\Sigma)$ whose nodes are the sorts in Σ, and such that $G(\Sigma)$ contains a labeled edge $S \xrightarrow{f} S'$ if and only if Σ contains a function symbol whose source sort is S and whose target sort is S'. We say that Σ is *acyclic* if $G(\Sigma)$ is so.

Proposition 2. *A DB theory T has decidable constraint satisfiability problem and admits a model completion in case it is axiomatized by finitely many universal one-variable formulae and Σ is acyclic.* $\triangleleft$

The proof is given in [10]. Since acyclicity of Σ yields local finiteness, we immediately get as a Corollary the decidability of safety problems for transitions systems based on DB schema satisfying the hypotheses of the above theorem.

5 Covers via Constrained Superposition

Of course, a model completion may not exist at all; Proposition 2 shows that it exists in case T is a DB theory axiomatized by universal one-variable formulae and Σ is acyclic. The second hypothesis is unnecessarily restrictive and the algorithm for quantifier elimination suggested by the proof of Proposition 2 is highly impractical: for this reason we are trying a different approach. In this section, we drop the acyclicity hypothesis and examine the case where the theory T is empty and the signature Σ may contain function symbols of any arity. Covers in this context were shown to exist already in [25], using an algorithm that, very roughly speaking, determines all the conditional equations that can be derived concerning the nodes of the congruence closure graph. An algorithm for the generation of interpolants, still relying on congruence closure [28] and similar to the one presented in [25], is supplied in [29].

We follow a different plan and we want to produce covers (and show that they exist) using *saturation-based theorem proving*. The natural idea to proceed in this sense is to take the matrix $\phi(\underline{e}, \underline{y})$ of the primitive formula $\exists \underline{e}\, \phi(\underline{e}, \underline{y})$ we want to compute the cover of: this is a conjunction of literals, so we consider each variable as a free constant, we saturate the corresponding set of ground literals and finally we output the literals involving only the $\underline{y}$. For saturation, one can use any version of the superposition calculus [38]. This procedure however for our problem is not sufficient. As a trivial counterexample consider the primitive formula $\exists e\,(R(e, y_1) \wedge \neg R(e, y_2))$: the set of literals $\{R(e, y_1), \neg R(e, y_2)\}$ is saturated (recall that we view e, y_1, y_2 as constants), however the formula has a non-trivial cover $y_1 \neq y_2$ which is *not* produced by saturation. If we move to signatures with function symbols, the situation is even worse: the set of literals $\{f(e, y_1) = y_1', f(e, y_2) = y_2'\}$ is saturated but the formula $\exists e\,(f(e, y_1) = y_1' \wedge f(e, y_2) = y_2')$ has the *conditional equality* $y_1 = y_2 \rightarrow y_1' = y_2'$ as cover. *Disjunctions of disequations* might also arise: the cover of $\exists e\, h(e, y_1, y_2) \neq h(e, y_1', y_2')$ (as well as the cover of $\exists e\, f(f(e, y_1), y_2) \neq f(f(e, y_1'), y_2')$, see Example 1 below) is $y_1 \neq y_1' \vee y_2 \neq y_2'$. [4]

Notice that our problem is different from the problem of producing ordinary quantifier-free interpolants via saturation based theorem proving [30]: for ordinary Craig interpolants, we have as input *two* quantifier-free formulae $\phi(\underline{e}, \underline{y}), \phi'(\underline{y}, \underline{z})$ such that $\phi(\underline{e}, \underline{y}) \rightarrow \phi'(\underline{y}, \underline{z})$ is valid; here we have a *single* formula $\phi(\underline{e}, \underline{y})$ in input and we are asked to find an interpolant which is good *for all possible* $\phi'(\underline{y}, \underline{z})$ such that $\phi(\underline{e}, \underline{y}) \rightarrow \phi'(\underline{y}, \underline{z})$ is valid. Ordinary interpolants can be extracted from a refutation of $\phi(\underline{e}, \underline{y}) \wedge \neg\phi'(\underline{y}, \underline{z})$, here we are not given any refutation at all (and we are not even supposed to find one).

What we are going to show is that, nevertheless, saturation via superposition can be used to produce covers, if suitably adjusted. In this section we consider signatures with n-ary function symbols (for all $n \geq 1$). For simplicity, we omit

[4] This example points out a problem that needs to be fixed in the algorithm presented in [25]: that algorithm in fact outputs only equalities, conditional equalities and single disequalities, so it cannot correctly handle this example.

n-ary relation symbols (you can easily handle them by rewriting $R(t_1, \dots, t_n)$ as $R(t_1, \dots, t_n) = true$, as customary in the paramodulation literature [38]).

We are going to compute the cover of a primitive formula $\exists \underline{e}\, \phi(\underline{e}, \underline{y})$ to be fixed for the remainder of this section. We call variables $\underline{e}$ *existential* and variables $\underline{y}$ *parameters*. By applying abstraction steps, we can assume that ϕ is *primitive flat*. i.e. that it is a conjunction of $\underline{e}$*-flat literals*, defined below. [By an abstraction step we mean replacing $\exists \underline{e}\, \phi$ with $\exists \underline{e}\, \exists e'(e' = u \wedge \phi')$, where e' is a fresh variable and ϕ' is obtained from ϕ by replacing some occurrences of a term $u(\underline{e}, \underline{y})$ by e'].

A term or a formula are said to be $\underline{e}$-free iff the existential variables do not occur in it. An $\underline{e}$-flat term is an $\underline{e}$-free term $t(\underline{y})$ or a variable from $\underline{e}$ or again it is of the kind $f(u_1, \dots, u_n)$, where f is a function symbol and $u_1, \dots, u_n$ are $\underline{e}$-free terms or variables from $\underline{e}$. An $\underline{e}$-flat literal is a literal of the form

$$t = a, \quad a \neq b$$

where t is an $\underline{e}$-flat term and a, b are either $\underline{e}$-free terms or variables from $\underline{e}$.

We assume the reader is familiar with standard conventions used in rewriting and paramodulation literature: in particular $s_{|p}$ denotes the subterm of s in position p and $s[u]_p$ denotes the term obtained from s by replacing $s_{|p}$ with u. We use $\equiv$ to indicate coincidence of syntactic expressions (as strings) to avoid confusion with equality symbol; when we write equalities like $s = t$ below, we may mean both $s = t$ or $t = s$ (an equality is seen as a multiset of two terms). For information on reduction ordering, see for instance [2].

We first replace variables $\underline{e} = e_1, \dots, e_n$ and $\underline{y} = y_1, \dots, y_m$ by free constants - we keep the names $e_1, \dots, e_n, y_1, \dots, y_m$ for these constants. Choose a reduction ordering $>$ total for ground terms such that $\underline{e}$-flat literals $t = a$ are always oriented from left to right in the following two cases: (i) t is not $\underline{e}$-free and a is $\underline{e}$-free; (ii) t is not $\underline{e}$-free, it is not equal to any of the $\underline{e}$ and a is a variable from $\underline{e}$. To obtain such properties, one may for instance choose a suitable Knuth-Bendix ordering taking weights in some transfinite ordinal, see [32].

Given two $\underline{e}$-flat terms t, u, we indicate with $E(t, u)$ the following procedure:

- $E(t, u)$ fails if t is $\underline{e}$-free and u is not $\underline{e}$-free (or vice versa);
- $E(t, u)$ fails if $t \equiv e_i$ and (either $t \equiv f(t_1, \dots, t_k)$ or $u \equiv e_j$ for $i \neq j$);
- $E(t, u) = \emptyset$ if $t \equiv u$;
- $E(t, u) = \{t = u\}$ if t and u are different but both $\underline{e}$-free;
- $E(t, u)$ fails if none of t, u is $\underline{e}$-free, $t \equiv f(t_1, \dots, t_k)$ and $u \equiv g(u_1, \dots, u_l)$ for $f \not\equiv g$;
- $E(t, u) = E(t_1, u_1) \cup \dots \cup E(t_k, u_k)$ if none of t, u is $\underline{e}$-free, $t \equiv f(t_1, \dots, t_k)$, $u \equiv f(u_1, \dots, u_k)$ and none of the $E(t_i, u_i)$ fails.

Notice that, whenever $E(t, u)$ succeeds, the formula $\bigwedge E(t, u) \to t = u$ is universally valid. The definition of $E(t, u)$ is motivated by the next lemma.

Lemma 2. *Let R be a convergent (i.e. terminating and confluent) ground rewriting system, whose rules consist of $\underline{e}$-free terms. Suppose that t and u are $\underline{e}$-flat terms with the same R-normal form. Then $E(t, u)$ does not fail and all pairs from $E(t, u)$ have the same R-normal form as well.* ◁

Proof. This is due to the fact that if t is not $\underline{e}$-free, no R-rewriting is possible at root position because rules from R are $\underline{e}$-free. $\dashv$

In the following, we handle *constrained* ground flat literals of the form $L \,\|\, C$ where L is a ground flat literal and C is a conjunction of ground equalities among $\underline{e}$-free terms. The logical meaning of $L \,\|\, C$ is the Horn clause $\bigwedge C \to L$.

In the literature, various calculi with constrained clauses were considered, starting e.g. from the non-ground constrained versions of the Superposition Calculus of [4,37]. The calculus we propose here is inspired by such versions and it has close similarities with a subcase of hierarchic superposition calculus [5], or rather to its "weak abstraction" variant from [6] (we thank an anonymous referee for pointing out this connection).

The rules of our *Constrained Superposition Calculus* follow; each rule applies provided the E subprocedure called by it does not fail. The symbol $\bot$ indicates the empty clause. Further explanations and restrictions to the calculus are given in the Remarks below.

Superposition Right (Constrained)
$$\frac{l = r \,\|\, C \qquad s = t \,\|\, D}{s[r]_p = t \,\|\, C \cup D \cup E(s_{|p}, l)} \qquad \text{if } l > r \text{ and } s > t$$

Superposition Left (Constrained)
$$\frac{l = r \,\|\, C \qquad s \neq t \,\|\, D}{s[r]_p \neq t \,\|\, C \cup D \cup E(s_{|p}, l)} \qquad \text{if } l > r \text{ and } s > t$$

Reflexion (Constrained)
$$\frac{t \neq u \,\|\, C}{\bot \,\|\, C \cup E(t, u)}$$

Demodulation (Constrained)
$$\frac{L \,\|\, C, \qquad l = r \,\|D}{L[r]_p \,\|\, C} \qquad \text{if } l > r,\ L_{|p} \equiv l \text{ and } C \supseteq D$$

Remark 1. The first three rules are inference rules: they are non-deterministically selected for application, until no rule applies anymore. The selection strategy for the rule to be applied is not relevant for the correctness and completeness of the algorithm (some variant of a 'given clause algorithm' can be applied). An inference rule *is not applied in case one premise is* $\underline{e}$*-free* (we have no reason to apply inferences to $\underline{e}$-free premises, since we are not looking for a refutation). $\triangleleft$

Remark 2. The Demodulation rule is a simplification rule: its application not only adds the conclusion to the current set of constrained literals, but it also removes the first premise. It is easy to see (e.g., representing literals as multisets of terms and extending the total reduction ordering to multisets), that one cannot have an infinite sequence of consecutive applications of Demodulation rules. $\triangleleft$

Remark 3. The calculus takes $\{L \| \emptyset \mid L \text{ is a flat literal from the matrix of } \phi\}$ as the initial set of constrained literals. It terminates when a *saturated* set of constrained literals is reached. We say that S is saturated iff every constrained literal that can be produced by an inference rule, after being exhaustively simplified via Demodulation, is already in S (there are more sophisticated notions of 'saturation up to redundancy' in the literature, but we do not need them). When it reaches a saturated set S, the algorithm outputs the conjunction of the clauses $\bigwedge C \to L$, varying $L \,\|\, C$ among the $\underline{e}$-free constrained literals from S. $\triangleleft$

We need some rule application policy to ensure termination: without any such policy, a set like $\{e = y \,\|\, \emptyset, f(e) = e \| \, \emptyset\}$ may produce by Right Superposition the infinitely many literals (all oriented from right to left) $f(y) = e \,\|\, \emptyset$, $f(f(y)) = e \,\|\, \emptyset$, $f(f(f(y))) = e \,\|\, \emptyset$, etc. The next Remark explains the policy we follow.

Remark 4. First, we apply Demodulation *only in case the second premise is of the kind* $e_j = t(\underline{y}) \,\|D$*, where* t *is* $\underline{e}$*-free.* Demodulation rule is applied with higher priority with respect to the inference rules. Inside all possible applications of Demodulation rule, we give priority to the applications where *both premises have the form* $e_j = t(\underline{y}) \,\|D$ (for the same e_j but with possibly different D's - the D from the second premise being included in the D of the first). In case we have two constrained literals of the kind $e_j = t_1(\underline{y}) \,\|D$, $e_j = t_2(\underline{y}) \,\|D$ inside our current set of constrained literals (notice that the e_j's and the D's here are the same), among the two possible applications of the Demodulation rule, we apply the rule that keeps the smallest t_i. Notice that in this way two different constrained literals cannot simplify each other. $\lhd$

We say that a constrained literal $L \,\|\, C$ belonging to a set of constrained literals S is *simplifiable in* S iff it is possible to apply (according to the above policy) a Demodulation rule removing it. A first effect of our policy is:

Lemma 3. *If a constrained literal* $L \,\|\, C$ *is simplifiable in* S*, then after applying to* S *any sequence of rules, it remains simplifiable until it gets removed. After being removed, if it is regenerated, it is still simplifiable and so it is eventually removed again.* $\lhd$

Proof. Suppose that $L \,\|\, C$ can be simplified by $e = t \,\|\, D$ and suppose that a rule is applied to the current set of constrained literals. Since there are simplifiable constrained literals, that rule cannot be an inference rule by the priority stated in Remark 4. For simplification rules, keep in mind again Remark 4. If $L \,\|\, C$ is simplified, it is removed; if none of $L \,\|\, C$ and $e = t \,\|\, D$ get simplified, the situation does not change; if $e = t \,\|\, D$ gets simplified, this can be done by some $e = t' \| \, D'$, but then $L \,\|\, C$ is still simplifiable - although in a different way - using $e = t' \| \, D'$ (we have that D' is included in D, which is in turn included in C). Similar observations apply if $L \,\|\, C$ is removed and re-generated. $\dashv$

Due to the above Lemma, if we show that a derivation (i.e. a sequence of rule applications) can produce terms only from a finite set, it is clear that when no new constrained literal is produced, saturation is reached. First notice that

Lemma 4. *Every constrained literal* $L \,\|\, C$ *produced during the run of the algorithm is* $\underline{e}$*-flat.* $\lhd$

Proof. The constrained literals from initialization are $\underline{e}$-flat. The Demodulation rule, applied according to Remark 4, produces an $\underline{e}$-flat literal out of an $\underline{e}$-flat literal. The same happens for the Superposition rules: in fact, since both the terms s and l from these rules are $\underline{e}$-flat, a Superposition may take place at root position or may rewrite some $l \equiv e_j$ with $r \equiv e_i$ or with $r \equiv t(\underline{y})$. $\dashv$

There are in principle infinitely many $\underline{e}$-flat terms that can be generated out of the $\underline{e}$-flat terms occurring in ϕ (see the above counterexample). We show however that only finitely many $\underline{e}$-flat terms can in fact occur during saturation and that one can determine in advance the finite set they are taken from.

To formalize this idea, let us introduce a hierarchy of $\underline{e}$-flat terms. Let D_0 be the $\underline{e}$-flat terms occurring in ϕ and let D_{k+1} be the set of $\underline{e}$-flat terms obtained by simultaneous rewriting of an $\underline{e}$-flat term from $\bigcup_{i\leq k} D_i$ via rewriting rules of the kind $e_j \to t_j(\underline{y})$ where the t_j are $\underline{e}$-flat $\underline{e}$-free terms from $\bigcup_{i\leq k} D_i$. The *degree* of an $\underline{e}$-flat term is the minimum k such that it belongs to set D_k (it is necessary to take the minimum because the same term can be obtained in different stages and via different rewritings).[5]

Lemma 5. *Let the $\underline{e}$-flat term t' be obtained by a rewriting $e_j \to u(\underline{y})$ from the $\underline{e}$-flat term t; then, if t has degree $k > 1$ and u has degree at most $k-1$, we have that t' has degree at most k.* $\lhd$

Proof. This is clear, because at the k-stage one can directly produce t' instead of just t: in fact, all rewriting producing directly t' replace an occurrence of some e_i by an $\underline{e}$-free term, so they are all done in parallel positions. $\dashv$

Proposition 3. *The saturation of the initial set of $\underline{e}$-flat constrained literals always terminates after finitely many steps.* $\lhd$

Proof. We show that all $\underline{e}$-flat terms that may occur during saturation have at most degree n (where n is the cardinality of $\underline{e}$). This shows that the saturation must terminate, because only finitely many terms may occur in a derivation (see the above observations). Let the algorithm during saturation reach the status S; we say that *a constraint C allows the explicit definition of e_j in S* iff S contains a constrained literal of the kind $e_j = t(\underline{y}) \parallel D$ with $D \subseteq C$. Now we show by mutual induction two facts concerning a constrained literal $L \parallel C \in S$:

(1) if an $\underline{e}$-flat term u of degree k occurs in L, then C allows the explicit definition of k different e_j in S;
(2) if L is of the kind $e_i = t(\underline{y})$, for an $\underline{e}$-flat $\underline{e}$-free term t of degree k, then either $e_i = t \parallel C$ can be simplified in S or C allows the explicit definition of $k+1$ different e_j in S (e_i itself is of course included among these e_j).

Notice that (1) is sufficient to exclude that any $\underline{e}$-flat term of degree bigger than n can occur in a constrained literal arising during the saturation process.

We prove (1) and (2) by induction on the length of the derivation leading to $L \parallel C \in S$. Notice that it is sufficient to check that (1) and (2) hold for the first time where $L \parallel C \in S$ because if C allows the explicit definition of a certain variable in S, it will continue to do so in any S' obtained from S by continuing the derivation (the definition may be changed by the Demodulation rule, but the fact that e_i is explicitly defined is forever). Also, by Lemma 3, a literal cannot become non simplifiable if it is simplifiable.

[5] Notice that, in the above definition of degree, constraints (attached to the rewriting rules occurring in our calculus) are ignored.

(1) and (2) are evident if S is the initial status. To show (1), suppose that u occurs for the first time in $L \parallel C$ as the effect of the application of a certain rule: we can freely assume that u does not occur in the literals from the premisses of the rule (otherwise induction trivially applies) and that u of degree k is obtained by rewriting *in a non-root position* some u' occurring in a constrained literal $L' \parallel D'$ via some $e_j \to t \parallel D$. This might be the effect of a Demodulation or Superposition in a non-root position (Superpositions in root position do not produce new terms). If u' has degree k, then by induction D' contains the required k explicit definitions, and we are done because D' is included in C. If u' has lower degree, then t must have degree at least $k-1$ (otherwise u does not reach degree k by Lemma 5). Then by induction on (2), the constraint D (also included in C) has $(k-1)+1 = k$ explicit definitions (when a constraint $e_j \to t \parallel D$ is selected for Superposition or for making Demodulations in a non-root position, it is itself not simplifiable according to the procedure explained in Remark 4).

To show (2), we analyze the reasons why the non simplifiable constrained literal $e_i = t(\underline{y}) \parallel C$ is produced (let k be the degree of t). Suppose it is produced from $e_i = u' \parallel C$ via Demodulation with $e_j = u(\underline{y}) \parallel D$ (with $D \subseteq C$) in a non-root position; if u' has degree at least k, we apply induction for (1) to $e_i = u' \parallel C$: by such induction hypotheses, we get k explicit definitions in C and we can add to them the further explicit definition $e_i = t(\underline{y})$ (the explicit definitions from C cannot concern e_i because $e_i = t(\underline{y}) \parallel C$ is not simplifiable). Otherwise, u' has degree less than k and u has degree at least $k-1$ by Lemma 5 (recall that t has degree k): by induction, $e_j = u \parallel D$ is not simplifiable (it is used as the active part of a Demodulation in a non-root position, see Remark 4) and supplies k explicit definitions, inherited by $C \supseteq D$. Note that e_i cannot have a definition in D, otherwise $e_i = t(\underline{y}) \parallel C$ would be simplifiable, so with $e_i = t(\underline{y}) \parallel C$ we get the required $k+1$ definitions.

The remaining case is when $e_i = t(\underline{y}) \parallel C$ is produced via Superposition Right. Such a Superposition might be at root or at a non-root position. We first analyse the case of a root position. This might be via $e_j = e_i \parallel C_1$ and $e_j = t(\underline{y}) \parallel C_2$ (with $e_j > e_i$ and $C = C_1 \cup C_2$ because $E(e_j, e_j) = \emptyset$), but in such a case one can easily apply induction. Otherwise, we have a different kind of Superposition at root position: $e_i = t(\underline{y}) \parallel C$ is obtained from $s = e_i \parallel C_1$ and $s' = t(\underline{y}) \parallel C_2$, with $C = C_1 \cup C_2 \cup E(s, s')$. In this case, by induction for (1), C_2 supplies k explicit definitions, to be inherited by C. Among such definitions, there cannot be an explicit definition of e_i otherwise $e_i = t(\underline{y}) \parallel C$ would be simplifiable, so again we get the required $k+1$ definitions.

In case of a Superposition at a non root-position, we have that $e_i = t(\underline{y}) \parallel C$ is obtained from $u' = e_i \parallel C_1$ and $e_j = u(\underline{y}) \parallel C_2$, with $C = C_1 \cup C_2$; here t is obtained from u' by rewriting e_j to u. This case is handled similarly to the case where $e_i = t(\underline{y}) \parallel C$ is obtained via Demodulation rule. $\dashv$

Having established termination, we now prove that our calculus computes covers; to this aim, we rely on refutational completeness of unconstrained Superposition Calculus (thus, our technique resembles the technique used [5,6] in order to prove refutational completeness of hierarchic superposition, although it is not clear whether Theorem 2 below can be derived from the results concerning hierarchic superposition - we are not just proving refutational completeness and we need to build proper superstructures):

Theorem 2. *Suppose that the above algorithm, taking as input the primitive* $\underline{e}$*-flat formula* $\exists \underline{e}\, \phi(\underline{e}, \underline{y})$*, gives as output the quantifier-free formula* $\psi(\underline{y})$*. Then the latter is a cover of* $\exists \underline{e}\, \phi(\underline{e}, \underline{y})$*.* ◁

Proof. Let S be the saturated set of constrained literals produced upon termination of the algorithm; let $S = S_1 \cup S_2$, where S_1 contains the constrained literals in which the $\underline{e}$ do not occur and S_2 is its complement. Clearly $\exists \underline{e}\, \phi(\underline{e}, \underline{y})$ turns out to be logically equivalent to

$$\bigwedge_{L \,\|\, C \in S_1} (\bigwedge C \to L) \wedge \exists \underline{e} \bigwedge_{L \,\|\, C \in S_2} (\bigwedge C \to L)$$

so, as a consequence, in view of Lemma 1 it is sufficient to show that every model $\mathcal{M}$ satisfying $\bigwedge_{L \,\|\, C \in S_1} (\bigwedge C \to L)$ via an assignment $\mathcal{I}$ to the variables $\underline{y}$ can be embedded into a model $\mathcal{M}'$ such that for a suitable extension $\mathcal{I}'$ of $\mathcal{I}$ to the variables $\underline{e}$ we have that $(\mathcal{M}', \mathcal{I}')$ satisfies also $\bigwedge_{L \,\|\, C \in S_2} (\bigwedge C \to L)$.

Fix $\mathcal{M}, \mathcal{I}$ as above. The diagram $\Delta(\mathcal{M})$ of $\mathcal{M}$ is obtained as follows. We take one free constant for each element of the support of $\mathcal{M}$ (by Löwenheim-Skolem theorem you can keep $\mathcal{M}$ at most countable, if you like) and we put in $\Delta(\mathcal{M})$ all the literals of the kind $f(c_1, \ldots, c_k) = c_{k+1}$ and $c_1 \neq c_2$ which are true in $\mathcal{M}$ (here the c_i are names for the elements of the support of $\mathcal{M}$). Let R be the set of ground equalities of the form $y_i = c_i$, where c_i is the name of $\mathcal{I}(y_i)$. Extend our reduction ordering in the natural way (so that $y_i = c_i$ and $f(c_1, \ldots, c_k) = c_{k+1}$ are oriented from left to right). Consider now the set of clauses

$$\Delta(\mathcal{M}) \;\cup\; R \;\cup\; \{\bigwedge C \to L \mid (L \,\|\, C) \in S\} \tag{3}$$

(below, we distinguish the positive and the negative literals of $\Delta(\mathcal{M})$ so that $\Delta(\mathcal{M}) = \Delta^+(\mathcal{M}) \cup \Delta^-(\mathcal{M})$). We want to saturate the above set in the standard Superposition Calculus. Clearly the rewriting rules in R, used as reduction rules, replace everywhere y_i by c_i inside the clauses of the kind $\bigwedge C \to L$. At this point, the negative literals from the equality constraints all disappear: if they are true in $\mathcal{M}$, they $\Delta^+(\mathcal{M})$-normalize to trivial equalities $c_i = c_i$ (to be eliminated by standard reduction rules) and if they are false in $\mathcal{M}$ they become part of clauses subsumed by true inequalities from $\Delta^-(\mathcal{M})$. Similarly all the $\underline{e}$-free literals not coming from $\Delta(\mathcal{M}) \cup R$ get removed. Let $\tilde{S}$ be the set of survived literals involving the $\underline{e}$ (they are not constrained anymore and they are $\Delta^+(\mathcal{M}) \cup R$-normalized): we show that they cannot produce new clauses. Let in fact (π) be an inference from the Superposition Calculus [38] applying to them. Since no superposition with $\Delta(\mathcal{M}) \cup R$ is possible, this inference must involve only literals from $\tilde{S}$; suppose it produces a literal $\tilde{L}$ from the literals $\tilde{L}_1, \tilde{L}_2$ (coming via $\Delta^+(\mathcal{M}) \cup R$-normalization from $L_1 \,\|\, C_1 \in S$ and $L_2 \,\|\, C_2 \in S$) as parent clauses. Then, by Lemma 2, our constrained inferences produce a constrained literal $L \parallel C$ such that the clause $\bigwedge C \to L$ normalizes to $\tilde{L}$ via $\Delta^+(\mathcal{M}) \cup R$. Since S is saturated, the constrained literal $L \parallel C$, after simplification, belongs to S. Now simplifications via our Constrained Demodulation and $\Delta(\mathcal{M})^+ \cup R$-normalization commute (they work at parallel positions, see Remark 4), so the inference (π) is redundant because $\tilde{L}$ simplifies to a literal already in $\tilde{S} \cup \Delta(\mathcal{M})$.

Thus the set of clauses (3) saturates without producing the empty clause. By the completeness theorem of the Superposition Calculus [3,27,38] it has a model $\mathcal{M}'$. This $\mathcal{M}'$ by construction fits our requests by Robinson Diagram Lemma. $\dashv$

Theorem 2 also proves the existence of the model completion of EUF.

Example 1. We compute the cover of the primitive formula $\exists e\, f(f(e, y_1), y_2) \neq f(f(e, y_1'), y_2')$ (one more example, taken from [25], is analyzed in [10]). Flattening gives the set of literals $\{\ f(e, y_1) = e_1,\ f(e_1, y_2) = e_1',\ f(e, y_1') = e_2,\ f(e_2, y_2') = e_2',\ e_1' \neq e_2'\ \}$. Superposition Right produces the constrained literal $e_1 = e_2 \parallel \{y_1 = y_1'\}$; supposing that we have $e_1 > e_2$, Superposition Right gives first $f(e_2, y_2) = e_1' \parallel \{y_1 = y_1'\}$ and then also $e_1' = e_2' \parallel \{y_1 = y_1', y_2 = y_2'\}$. Superposition Left and Reflexion now produce $\bot \parallel \{y_1 = y_1', y_2 = y_2'\}$. Thus the clause $y_1 = y_1' \wedge y_2 = y_2' \to \bot$ will be part of the output (actually, this will be the only clause in the output). $\triangleleft$

In the special case where the signature Σ contains only unary function symbols, only empty constraints can be generated; in case Σ contains also relation symbols of arity $n > 1$, the only constrained clauses that can be generated have the form $\bot \parallel \{t_1 = t_1', \dots, t_{n-1} = t_{n-1}'\}$. Also, it is not difficult to see that in a derivation at most one explicit definition $e_i = t(\underline{y}) \parallel \emptyset$ can occur for every e_i: as soon as this definition is produced, all occurrences of e_i are rewritten to t. This shows that Constrained Superposition computes covers in polynomial time for the empty theory, whenever the signature Σ matches the restrictions of Definition 2 for DB schemata. More details on complexity are given in [10] (where a quadratic bound is obtained).

6 Conclusions and Future Work

As evident from Subsect. 4.1, our main motivation for investigating covers originated from the verification of data-aware processes. Such applications require database (DB) signatures to contain only unary function symbols (besides relations of every arity). We observed that computing covers of primitive formulae in such signatures requires only polynomial time. In addition, if relation symbols are at most binary, *the cover of a primitive formula is a conjunction of literals*: this is crucial in applications, because model checkers like MCMT [21] and CUBICLE [14] represent sets of reachable states as primitive formulae. This makes cover computations a quite attractive technique in database-driven model checking.

Our cover algorithm for DB signatures has been implemented in the model checker MCMT. A first experimental evaluation (based on the existing benchmark provided in [31], which samples 32 real-world BPMN workflows taken from the BPMN official website http://www.bpmn.org/) is described in [11]. The benchmark set is available as part of the last distribution 2.8 of MCMT http://users.mat.unimi.it/users/ghilardi/mcmt/ (see the subdirectory `/examples/dbdriven` of the distribution). The user manual, also included in the distribution, contains a dedicated section giving essential information on how to encode relational artifact systems (comprising both first order and second order variables) in MCMT specifications and how to produce user-defined examples in the database driven framework. Although an extensive experimentation is outside the focus of this paper, we mention that the first experiments were very encouraging: the tool was able to solve in few seconds all the proposed benchmarks and the cover computations generated automatically during model-checking search were discharged instantaneously.

This experimental setup motivates new research to extend Proposition 2 to further theories axiomatizing integrity constraints used in DB applications. Combined cover algorithms (along the perspectives in [25]) could be crucial also in this setting. Practical algorithms for the computation of covers in the theories falling under the hypotheses of Proposition 2 need to be designed: as a little first example, in [10] we show how to handle Axiom (2) by light modifications to our techniques. Symbol elimination of function and predicate variables should also be combined with cover computations.

Acknowledgements. This research has been partially supported by the UNIBZ CRC projects *REKAP: Reasoning and Enactment for Knowledge-Aware Processes* and *PWORM: Planning for Workflow Management.*

References

1. Baader, F., Ghilardi, S., Tinelli, C.: A new combination procedure for the word problem that generalizes fusion decidability results in modal logics. Inf. Comput. **204**(10), 1413–1452 (2006)
2. Baader, F., Nipkow, T.: Term Rewriting and All That. Cambridge University Press, Cambridge (1998)
3. Bachmair, L., Ganzinger, H.: Rewrite-based equational theorem proving with selection and simplification. J. Log. Comput. **4**(3), 217–247 (1994)
4. Bachmair, L., Ganzinger, H., Lynch, C., Snyder, W.: Basic paramodulation. Inf. Comput. **121**(2), 172–192 (1995)
5. Bachmair, L., Ganzinger, H., Waldmann, U.: Refutational theorem proving for hierarchic first-order theories. Appl. Algebra Eng. Commun. Comput. **5**, 193–212 (1994)
6. Baumgartner, P., Waldmann, U.: Hierarchic superposition with weak abstraction. In: Bonacina, M.P. (ed.) CADE 2013. LNCS (LNAI), vol. 7898, pp. 39–57. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-38574-2_3
7. Bojańczyk, M., Segoufin, L., Toruńczyk, S.: Verification of database-driven systems via amalgamation. In: Proceedings of PODS, pp. 63–74 (2013)
8. Bruttomesso, R., Ghilardi, S., Ranise, S.: Quantifier-free interpolation in combinations of equality interpolating theories. ACM Trans. Comput. Log. **15**(1), 5:1–5:34 (2014)

9. Calvanese, D., De Giacomo, G., Montali, M.: Foundations of data aware process analysis: a database theory perspective. In: Proceedings of PODS (2013)
10. Calvanese, D., Ghilardi, S., Gianola, A., Montali, M., Rivkin, A.: Quantifier elimination for database driven verification. CoRR, abs/1806.09686 (2018)
11. Calvanese, D., Ghilardi, S., Gianola, A., Montali, M., Rivkin, A.: Verification of data-aware processes via array-based systems (extended version). Technical report arXiv:1806.11459, arXiv.org (2018)
12. Calvanese, D., Ghilardi, S., Gianola, A., Montali, M., Rivkin, A.: From model completeness to verification of data aware processes. In: Lutz, C., Sattler, U., Tinelli, C., Turhan, A.Y., Wolter, F. (eds.) Description Logic, Theory Combination, and All That. LNCS, vol. 11560, pp. 212–239. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-22102-7_10
13. Chang, C.-C., Keisler, J.H.: Model Theory, 3rd edn. North-Holland Publishing Co., Amsterdam (1990)
14. Conchon, S., Goel, A., Krstić, S., Mebsout, A., Zaïdi, F.: Cubicle: a parallel SMT-based model checker for parameterized systems. In: Madhusudan, P., Seshia, S.A. (eds.) CAV 2012. LNCS, vol. 7358, pp. 718–724. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-31424-7_55
15. Deutsch, A., Hull, R., Patrizi, F., Vianu, V.: Automatic verification of data-centric business processes. In: Proceedings of ICDT, pp. 252–267 (2009)
16. Deutsch, A., Li, Y., Vianu, V.: Verification of hierarchical artifact systems. In: Proceedings of PODS, pp. 179–194. ACM Press (2016)
17. Ghilardi, S.: Model theoretic methods in combined constraint satisfiability. J. Autom. Reason. **33**(3–4), 221–249 (2004)
18. Ghilardi, S., Gianola, A.: Interpolation, amalgamation and combination (the non-disjoint signatures case). In: Dixon, C., Finger, M. (eds.) FroCoS 2017. LNCS (LNAI), vol. 10483, pp. 316–332. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-66167-4_18
19. Ghilardi, S., Gianola, A.: Modularity results for interpolation, amalgamation and superamalgamation. Ann. Pure Appl. Log. **169**(8), 731–754 (2018)
20. Ghilardi, S., Nicolini, E., Zucchelli, D.: A comprehensive combination framework. ACM Trans. Comput. Log. **9**(2), 54 p. (2008). Article no. 8
21. Ghilardi, S., Ranise, S.: MCMT: a model checker modulo theories. In: Giesl, J., Hähnle, R. (eds.) IJCAR 2010. LNCS (LNAI), vol. 6173, pp. 22–29. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-14203-1_3
22. Ghilardi, S., van Gool, S.J.: Monadic second order logic as the model companion of temporal logic. In: Proceedings of LICS, pp. 417–426 (2016)
23. Ghilardi, S., van Gool, S.J.: A model-theoretic characterization of monadic second order logic on infinite words. J. Symb. Log. **82**(1), 62–76 (2017)
24. Ghilardi, S., Zawadowski, M.: Sheaves, Games, and Model Completions: A Categorical Approach to Nonclassical Propositional Logics. Trends in Logic-Studia Logica Library, vol. 14. Kluwer Academic Publishers, Dordrecht (2002)
25. Gulwani, S., Musuvathi, M.: Cover algorithms and their combination. In: Drossopoulou, S. (ed.) ESOP 2008. LNCS, vol. 4960, pp. 193–207. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-78739-6_16
26. Hoder, K., Bjørner, N.: Generalized property directed reachability. In: Cimatti, A., Sebastiani, R. (eds.) SAT 2012. LNCS, vol. 7317, pp. 157–171. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-31612-8_13
27. Hsiang, J., Rusinowitch, M.: Proving refutational completeness of theorem-proving strategies: the transfinite semantic tree method. J. ACM **38**(3), 559–587 (1991)

28. Kapur, D.: Shostak's congruence closure as completion. In: Comon, H. (ed.) RTA 1997. LNCS, vol. 1232, pp. 23–37. Springer, Heidelberg (1997). https://doi.org/10.1007/3-540-62950-5_59
29. Kapur, D.: Nonlinear polynomials, interpolants and invariant generation for system analysis. In: Proceedings of the 2nd International Workshop on Satisfiability Checking and Symbolic Computation Co-Located with ISSAC (2017)
30. Kovács, L., Voronkov, A.: Interpolation and symbol elimination. In: Schmidt, R.A. (ed.) CADE 2009. LNCS (LNAI), vol. 5663, pp. 199–213. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-02959-2_17
31. Li, Y., Deutsch, A., Vianu, V.: VERIFAS: a practical verifier for artifact systems. PVLDB **11**(3), 283–296 (2017)
32. Ludwig, M., Waldmann, U.: An extension of the knuth-bendix ordering with LPO-like properties. In: Dershowitz, N., Voronkov, A. (eds.) LPAR 2007. LNCS (LNAI), vol. 4790, pp. 348–362. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-75560-9_26
33. McMillan, K.L.: Lazy abstraction with interpolants. In: Ball, T., Jones, R.B. (eds.) CAV 2006. LNCS, vol. 4144, pp. 123–136. Springer, Heidelberg (2006). https://doi.org/10.1007/11817963_14
34. Nicolini, E., Ringeissen, C., Rusinowitch, M.: Data structures with arithmetic constraints: a non-disjoint combination. In: Ghilardi, S., Sebastiani, R. (eds.) FroCoS 2009. LNCS (LNAI), vol. 5749, pp. 319–334. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-04222-5_20
35. Nicolini, E., Ringeissen, C., Rusinowitch, M.: Satisfiability procedures for combination of theories sharing integer offsets. In: Kowalewski, S., Philippou, A. (eds.) TACAS 2009. LNCS, vol. 5505, pp. 428–442. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-00768-2_35
36. Nicolini, E., Ringeissen, C., Rusinowitch, M.: Combining satisfiability procedures for unions of theories with a shared counting operator. Fundam. Inform. **105**(1–2), 163–187 (2010)
37. Nieuwenhuis, R., Rubio, A.: Theorem proving with ordering and equality constrained clauses. J. Symb. Comput. **19**(4), 321–351 (1995)
38. Nieuwenhuis, R., Rubio, A.: Paramodulation-based theorem proving. In: Handbook of Automated Reasoning, vol. 2, pp. 371–443. MIT Press (2001)
39. Pitts, A.M.: On an interpretation of second order quantification in first order intuitionistic propositional logic. J. Symb. Log. **57**(1), 33–52 (1992)
40. Rybina, T., Voronkov, A.: A logical reconstruction of reachability. In: Broy, M., Zamulin, A.V. (eds.) PSI 2003. LNCS, vol. 2890, pp. 222–237. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-39866-0_24
41. Sofronie-Stokkermans, V.: On interpolation and symbol elimination in theory extensions. In: Olivetti, N., Tiwari, A. (eds.) IJCAR 2016. LNCS (LNAI), vol. 9706, pp. 273–289. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-40229-1_19
42. Sofronie-Stokkermans, V.: On interpolation and symbol elimination in theory extensions. Log. Methods Comput. Sci. **14**(3), 1–41 (2018)
43. Vianu, V.: Automatic verification of database-driven systems: a new frontier. In: Proceedings of ICDT, pp. 1–13 (2009)
44. Wheeler, W.H.: Model-companions and definability in existentially complete structures. Isr. J. Math. **25**(3–4), 305–330 (1976)