On the Fixpoint Theory of Equality and Its Applications*

Andrzej Szałas[1,2] and Jerzy Tyszkiewicz[3]

[1] Dept. of Computer and Information Science, Linköping University
SE-581 83 Linköping, Sweden
andsz@ida.liu.se
[2] The University of Economics and Computer Science, Olsztyn, Poland
[3] Institute of Informatics, University of Warsaw, ul. Banacha 2, 02-097 Warsaw, Poland
jty@mimuw.edu.pl

Abstract. In the current paper we first show that the fixpoint theory of equality is decidable. The motivation behind considering this theory is that second-order quantifier elimination techniques based on a theorem given in [16], when successful, often result in such formulas. This opens many applications, including automated theorem proving, static verification of integrity constraints in databases as well as reasoning with weakest sufficient and strongest necessary conditions.

1 Introduction

In this paper we investigate the fixpoint theory of equality, FEQ, i.e., the classical first-order theory with equality as the only relation symbol, extended by allowing least and greatest fixpoints. We show that FEQ is decidable.

The motivation behind considering this theory follows from important applications naturally appearing in artificial intelligence and databases. Namely, we propose a technique, which basically depends on expressing some interesting properties as second-order formulas with all relation symbols appearing in the scope of second-order quantifiers, then on eliminating second-order quantifiers, if possible, and obtaining formulas expressed in the theory FEQ and finally, on reasoning in FEQ.

Second-order formalisms are frequent in knowledge representation. On the other hand, second-order logic is too complex[1] to be directly applied in practical reasoning. The proposed technique allows one to reduce second-order reasoning to fixpoint calculus for a large class of formulas and then to apply the decision procedure for FEQ.

To achieve our goal we first introduce a logic with simultaneous least fixpoints (Section 2) and then define the theory FEQ, prove its decidability and estimate complexity of reasoning (see Section 3). Next, in Section 4, we recall the fixpoint theorem of [16]. Then we discuss some applications of the proposed technique in automated theorem proving (Section 5.1), static verification of integrity constraints in deductive databases

* Supported in part by the grants 3 T11C 023 29 and 4 T11C 042 25 of the Polish Ministry of Science and Information Society Technologies.

[1] It is totally undecidable over arbitrary models and PSPACE-complete over finite models.

R.A. Schmidt (Ed.): RelMiCS /AKA 2006, LNCS 4136, pp. 388–401, 2006.

(Section 5.2) and reasoning with weakest sufficient and strongest necessary conditions as considered in [12,7] (Section 5.3).

To our best knowledge, the method proposed in Section 5.1 is original. The method discussed in Section 5.2 substantially extends the method of [9] by allowing recursive rules in addition to relational databases considered in [9]. The method presented in Section 5.3 shows a uniform approach to various forms of reasoning important in many artificial intelligence applications.

2 Fixpoint Logic

In this paper we deal with classical first-order logic (FOL) and the simultaneous least fixpoint logic (SLFP) with equality as a logical symbol, i.e., whenever we refer to the empty signature, we still allow the equality symbol within formulas.

We assume that the reader is familiar with FOL and define below syntax and semantics of SLFP.

Many of the notions of interest for us are syntax independent, so the choice of a syntactical representation of a particular semantics of fixpoints is immaterial. In this semantical sense the logic we consider has been introduced by Chandra and Harel in [5,4]. However, here we use a different syntax. A number of different definitions of SLFP, though of the same expressive power, can be found in the literature. All of them allow iterating a FOL formula up to a fixpoint. The difference is in the form of iteration.

Definition 2.1. *A relation symbol R* occurs positively *(respectively* negatively*) in a formula A if it appears under an even (respectively odd) number of negations.*[2]

A formula A is positive w.r.t. relation symbol R *iff all occurrences of R in A are positive. A formula A is* negative w.r.t. relation symbol R *iff all occurrences of R in A are negative.* ◁

Definition 2.2. *Let $\varphi_i(R_1, \ldots, R_\ell, \bar{x}_i, \bar{y}_i)$, for $i = 1, \ldots, \ell$, be* FOL *formulas, where $\bar{x}_i$ and $\bar{y}_i$ are all free first-order variables of φ_i, $|\bar{x}_i| = k_i$, none of the x's is among the y's and where, for $i = 1, \ldots, \ell$, R_i are k_i-argument relation symbols, all of whose occurrences in $\varphi_1, \ldots, \varphi_\ell$ are positive. Then the formula*

$$\text{SLFP}\,[R_1(\bar{x}_1) \equiv \varphi_1(R_1, \ldots, R_\ell, \bar{x}_1, \bar{y}_1), \ldots, R_\ell(\bar{x}_\ell) \equiv \varphi_\ell(R_1, \ldots, R_\ell, \bar{x}_\ell, \bar{y}_\ell)]$$

is called a simultaneous fixpoint formula *(with variables $\bar{x}_1, \ldots, \bar{x}_\ell, \bar{y}_1 \ldots, \bar{y}_\ell$ free). In the rest of the paper we often abbreviate the above formula by* SLFP $[\bar{R} \equiv \bar{\varphi}]$.

Let σ be a signature. Then the set of SLFP formulas over σ *is inductively defined as the least set containing formulas of* FOL *over σ, closed under the usual syntax rules of first-order logic and applications of simultaneous least fixpoints.* ◁

Note that according to the above rules, the fixpoint operators cannot be nested in SLFP, however, it is permitted to use boolean combinations of fixpoints, as well as to quantify variables outside of them.

[2] It is assumed here that all implications of the form $p \rightarrow q$ are substituted by $\neg p \vee q$ and all equivalences of the form $p \equiv q$ are substituted by $(\neg p \vee q) \wedge (\neg q \vee p)$.

FOL^k and SLFP^k stand for the sets of those formulas in FOL and SLFP, respectively, in which only at most k distinct first-order variable symbols occur.

For a structure $\mathbb{A}$, by A we denote the domain of $\mathbb{A}$. By A^k we denote the cartesian product $\underbrace{A \times \ldots \times A}_{k-\text{times}}$. By ω we denote the set of natural numbers.

We assume the standard semantics of FOL. For SLFP we need a semantical rule concerning the semantics of the formula $\textsc{Slfp}\,[\bar{R} \equiv \bar{\varphi}]$.

Further on $\bar{x} : \bar{a}, R_1 : \Phi_1, \ldots, R_\ell : \Phi_\ell$ denotes a valuation assigning $\bar{a}$ to $\bar{x}$ and Φ_i to R_i, for $i = 1, \ldots, \ell$. The values of the remaining variables play the rôle of parameters and are not reflected in the notation.

Given a structure $\mathbb{A}$, we define the sequence $(\bar{\Phi}^\alpha) = (\langle \Phi_1^\alpha, \ldots, \Phi_\ell^\alpha \rangle)$ indexed by ordinals α, by the following rules,

$$\begin{array}{lll} \Phi_i^0 & = \emptyset \ \text{ for } i = 1, \ldots, \ell & \\ \Phi_i^{\alpha+1} & = \{\bar{b} \in A^{k_i} \mid \mathbb{A}, \bar{x}_i : \bar{b}, R_1 : \Phi_1^\alpha, \ldots, R_\ell : \Phi_\ell^\alpha \models \varphi_i\} & \text{for } i = 1, \ldots, \ell \\ \Phi_i^\alpha & = \bigcup_{\beta<\alpha} \Phi_i^\beta & \text{for } i = 1, \ldots, \ell, \text{ when } \alpha \text{ is a limit ordinal.} \end{array}$$

Since each φ_i is positive in all the R_j's, a simple transfinite induction shows that the sequence $(\bar{\Phi}^\alpha)$ is ascending in each of the coordinates.

Let $\bar{\Phi}^\infty \stackrel{\text{def}}{=} \langle \Phi_1^\infty, \ldots, \Phi_\ell^\infty \rangle = \left\langle \bigcup_\alpha \Phi_1^\alpha, \ldots, \bigcup_\alpha \Phi_\ell^\alpha \right\rangle$. Then we define

$\mathbb{A}, \bar{x}_i : \bar{a}_i \models \textsc{Slfp}\,[\bar{R} \equiv \bar{\varphi}]$ iff $\bar{a}_i \in \Phi_i^\infty$ for $i = 1, \ldots, \ell$.

3 Fixpoint Theory of Equality

3.1 The Main Results

Before proceeding, we introduce the main tools.

Below by $A(\bar{x})[\bar{t}]$ we mean the application of $A(\bar{x})$ to terms (or, dependently on the context, to domain values) $\bar{t}$.

Definition 3.1. *Let $\mathbb{A}, \mathbb{B}$ be two structures over a common signature. We write $\mathbb{A} \equiv_k \mathbb{B}$ iff $\mathbb{A}$ and $\mathbb{B}$ cannot be distinguished by any FOL^k sentence, i.e., when for every sentence φ of first-order logic with k variables, $\mathbb{A} \models \varphi$ iff $\mathbb{B} \models \varphi$.*

For two tuples $\bar{a} \in A^k$ and $\bar{b} \in B^k$ we write $\mathbb{A}, \bar{a} \equiv_k \mathbb{B}, \bar{b}$ iff those tuples cannot be distinguished by any FOL^k formula in $\mathbb{A}$ and $\mathbb{B}$, i.e., when for every formula $\varphi(\bar{x}) \in \mathrm{FOL}^k$, $\mathbb{A} \models \varphi[\bar{a}]$ iff $\mathbb{B} \models \varphi[\bar{b}]$. ◁

Another fact that we will need is a characterization of the expressive power of FOL^k in terms of an infinitary Ehrenfeucht-Fraïssé-style pebble game. This game characterizes the expressive power of the logic we have introduced in the sense formulated in Theorem 3.3 of [3,8,17].

Definition 3.2 (The Game).

Players, board and pebbles. *The game is played by two players, Spoiler and Duplicator, on two σ-structures $\mathbb{A}, \mathbb{A}'$ with two distinguished tuples $\bar{a} \in A^k$ and $\bar{a}' \in \mathbb{A}'$. There are k pairs of pebbles: $(1,1'), \ldots, (k,k')$. Pebbles without primes are intended to be placed on elements of A, while those with primes on elements of A'.*

Initial position. *Initially, the pebbles are located as follows: pebble i is located on a_i, and pebble i' is located on a'_i, for $i = 1, \ldots, k$.*

Moves. *In each of the moves of the game, Spoiler is allowed to choose one of the structures and one of the pebbles placed on an element of that structure and move it onto some other element of the same structure. Duplicator must place the other pebble from that pair on some element in the other structure so that the partial function from $\mathbb{A}$ to $\mathbb{A}'$ mapping $x \in \mathbb{A}$ on which pebble i is placed onto the element $x' \in \mathbb{A}'$ on which pebble i' is placed and constants in $\mathbb{A}$ onto the corresponding constants in $\mathbb{A}'$, is a partial isomorphism. Spoiler is allowed to alternate between the structures as often as he likes, when choosing elements.*

Who wins? *Spoiler wins if Duplicator does not have any move preserving the isomorphism. We say that Duplicator has a winning strategy if he can play forever despite of the moves of Spoiler, preventing him from winning.* ◁

Theorem 3.3. *Let $\mathbb{A}, \mathbb{B}$ be any two structures of a common signature. Then Duplicator has a winning strategy in the game on $\mathbb{A}, \bar{a}$ and $\mathbb{B}, \bar{b}$ iff $\mathbb{A}, \bar{a} \equiv_k \mathbb{B}, \bar{b}$.* ◁

Henceforth we restrict our attention to the theory and models of pure equality. Let for a cardinal number $\mathfrak{m}$ the symbol $\mathbb{E}_\mathfrak{m}$ stand for the only (up to isomorphism) model of pure equality of cardinality $\mathfrak{m}$.

The following theorem can easily be proved using Theorem 3.3.

Theorem 3.4. *Let $k \in \omega$. Then for any cardinal numbers $\mathfrak{m}, \mathfrak{n} \geq k$ and any two tuples $\bar{a}, \bar{b}$ of length k over $\mathbb{E}_\mathfrak{m}$ and $\mathbb{E}_\mathfrak{n}$, respectively, $\mathbb{E}_\mathfrak{m}, \bar{a} \equiv_k \mathbb{E}_\mathfrak{n}, \bar{b}$ if and only if for every $i, j \leq k$ the equivalence $a_i = a_j \equiv b_i = b_j$ holds.* ◁

Proof. By theorem 3.3 it suffices to prove that the Duplicator has a winning strategy in the game iff for every $i, j \leq k$, $a_i = a_j \equiv b_i = b_j$.

If the equivalence does not hold, then certainly the Duplicator lost already at the beginning. In turn, if it does, than the initial position has the required isomorphism, and this can be preserved by the Duplicator, since the structures have at least as many elements as the number of pebbles, so the Duplicator can mimic any move of the Spoiler. ◁

Henceforth if the equivalence $a_i = a_j \equiv b_i = b_j$ holds for every $i, j \leq k$ for two tuples $\bar{a}, \bar{b}$ of length k, we will write $\bar{a} \equiv_k \bar{b}$. Note that already in $\mathbb{E}_k$ there are tuples which are representatives of all the equivalence classes of $\equiv_k$.

Definition 3.5. *The* quantifier rank of a formula α, *denoted by $r(\alpha)$, is defined inductively by setting $r(\alpha) \stackrel{\text{def}}{=} 0$ when α contains no quantifiers, $r(\neg\alpha) \stackrel{\text{def}}{=} r(\alpha)$, for any binary propositional connective $\circ$, $r(\alpha \circ \beta) \stackrel{\text{def}}{=} \max\{r(\alpha), r(\beta)\}$ and $r(\exists\alpha) \stackrel{\text{def}}{=} r(\forall\alpha) \stackrel{\text{def}}{=} r(\alpha) + 1$.* ◁

An important result is the following theorem, provided in [10, Theorem 2.7].

Theorem 3.6. *Let $0 < n \in \omega$ and let all the first-order formulas φ_i in an* SLFP *formula* $\textsc{Slfp}\,[\bar{R} \equiv \bar{\varphi}]$ *have at most k free variables and be of quantifier rank at most d. Then, over the empty signature,*[3] *each component Φ_i^n is definable by a first-order formula with at most k variables and of quantifier rank at most dn, i.e., for any $0<n\in\omega$ there are formulas $\varphi_1^n, \ldots, \varphi_\ell^n$ of FOL^k of quantifier rank $\leq dn$ such that for any structure $\mathbb{A}$ over the empty signature, $\Phi_i^n = \{\bar{a} \in A^{k_i} \mid \mathbb{A}, \bar{x}_i : \bar{a} \models \varphi(\bar{x}_i)\}$.* ◁

Next, an application of Theorem 3.4 and the previous results, yields the following consequence.

Corollary 3.7. *Let $0 < k \in \omega$. If $\mathbb{A}$ is a model of pure equality of cardinality at least k, $\bar{a} \in A^k$, and $\varphi(\bar{x}) \in \mathrm{SLFP}^k$, then $\mathbb{A} \models \varphi[\bar{a}]$ iff $\mathbb{E}_k \models \varphi[\bar{a}']$, where $\bar{a}' \equiv_k \bar{a}$.*

Proof. First, we claim that every subformula of φ of the form $\textsc{Slfp}\,[\bar{R} \equiv \bar{\varphi}]$ can be substituted by an FOL^k formula, equivalent to the former both in $\mathbb{A}$ and $\mathbb{E}_k$.

Indeed, in $\mathbb{E}_k$ the sequence of stages $(\bar{\Phi}^\alpha) = (\langle \Phi_1^\alpha, \ldots, \Phi_\ell^\alpha \rangle)$ reaches a fixpoint in a finite number of iterations, say K, i.e., $\bar{\Phi}^\infty = \bar{\Phi}^K$. The reason is that this sequence is ascending in each of the coordinates, and each coordinate for each α is a subset of a fixed, finite set. Therefore

$$\mathbb{E}_k \models \bigwedge_{i=1}^{\ell} \forall \bar{x} \big(\varphi_i^K(\bar{x}) \equiv \varphi_i^{K+1}(\bar{x})\big),$$

where $\varphi_i^K(\bar{x})$ are the formulas from Theorem 3.6. $\mathbb{A}$ is isomorphic to some $\mathbb{E}_\mathfrak{m}$ for some $\mathfrak{m} \geq k$, so by Theorem 3.4,

$$\mathbb{A} \models \bigwedge_{i=1}^{\ell} \forall \bar{x} \big(\varphi_i^K(\bar{x}) \equiv \varphi_i^{K+1}(\bar{x})\big),$$

This sentence asserts that the iteration of $\textsc{Slfp}\,[\bar{R} \equiv \bar{\varphi}]$ stops in $\mathbb{A}$ after at most K steps, too. It is now routine to use the FOL^k formulas $\varphi_i^K(\bar{x})$ to replace $\textsc{Slfp}\,[\bar{R} \equiv \bar{\varphi}]$ in φ.

Our claim has been proven. So let $\varphi' \in \mathrm{FOL}^k$ be equivalent to φ in both $\mathbb{A}$ and $\mathbb{E}_k$, and obtained by the substitution of all fixpoints of φ by their FOL^k-equivalents.

Now by Theorem 3.4 it follows that $\mathbb{A} \models \varphi'[\bar{a}]$ iff $\mathbb{E}_k \models \varphi'[\bar{a}']$, where $\bar{a}' \equiv_k \bar{a}$, and this carries over to the formula φ, as desired. ◁

3.2 The Complexity

Now we turn to the problem of satisfiability of SLFP formulas over the empty signature. This means that still the only predicate allowed in formulas is the equality.

By the results of the previous section, we have the following equivalence:

Theorem 3.8. *A formula φ of SLFP^k is satisfiable if and only if it is satisfiable in one of the structures $\mathbb{E}_1, \ldots, \mathbb{E}_k$.*

[3] Recall that equality is still allowed, since it is a logical symbol.

Proof. Indeed, any structure over the empty signature is isomorphic to one of the form $\mathbb{E}_{\mathfrak{m}}$, and since $\mathbb{E}_{\mathfrak{m}} \equiv_k \mathbb{E}_k$, the equivalence follows. ◁

This suggests the following algorithm for testing satisfiability of fixpoint formulas over the empty signature: for a given formula $\varphi(\bar{x}) \in \mathrm{SLFP}^k$ we test if it is satisfied by $\langle \mathbb{A}, \bar{a} \rangle$, where $\mathbb{A}$ ranges over all (pure equality) structures of cardinality at most k, and $\bar{a}$ ranges over all equality types of vectors of length $|\bar{x}|$ of elements from A.

Concerning the complexity of this procedure, the number of structures to be tested is linear in k. The number of iterations of any fixpoint in SLFP^k is bounded by $O(B(k)^\ell)$, where $B(n)$ is the n-th Bell number and ℓ the maximal number of formulas whose simultaneous fixed point is used. Indeed, $B(k)$ is the number of $\equiv_k$-equivalence classes. Thus computing the fixpoints literally, according to the definition, takes time bounded by a polynomial of $B(k)^\ell$, and computing the first-order constructs increases this by only a polynomial factor.

Therefore the algorithm we obtained is of exponential complexity.

4 The Fixpoint Theorem

Further on we deal with the first- and the second-order classical logic with equality.

Below we recall the theorem for elimination of second-order quantifiers, proved in [16]. This theorem, combined with the decidability result given in Section 3.2, provides us with a powerful tool for deciding many interesting properties, as shown in Section 5. For an overview of the related techniques see [15].

Let $B(X)$ be a second-order formula, where X is a k-argument relational variable and let $C(\bar{x})$ be a first-order formula with free variables $\bar{x} = \langle x_1, \ldots, x_k \rangle$. Then by $B[X(\bar{t}) := C(\bar{x})]$ we mean the formula obtained from $B(X)$ by substituting each occurrence of X of the form $X(\bar{t})$ in $B(X)$ by $C(\bar{t})$, renaming the bound variables in $C(\bar{x})$ with fresh variables.

Example 4.1. Let $B(X) \equiv \forall z[X(y,z) \vee X(f(y), g(x,z))]$, where X is a relational variable and let $C(x,y) \equiv \exists z R(x,y,z)$. Then $B[X(t_1,t_2) := C(x,y)]$ is defined by

$$\forall z[\ \underbrace{\exists z' R(y,z,z')}_{C'(y,z)} \vee \underbrace{\exists z' R(f(y), g(x,z), z')}_{C'(f(y),g(x,z))}\ ],$$

where $C'(x,y)$ is obtained from $C(x,y)$ by renaming the bound variable z with z'. ◁

Recall that by $A(\bar{x})[\bar{t}]$ we mean the application of $A(\bar{x})$ to terms $\bar{t}$.

The following theorem, substantial for the applications considered in Section 5, has been provided in [16]. Below, for simplicity, we use the standard least and greatest fixpoint operators LFP and GFP rather than simultaneous fixpoints.

Theorem 4.2. *Assume that formula A is a first-order formula positive w.r.t. X.*

– *if B is a first-order formula negative w.r.t. X then*

$$\exists X \forall \bar{y}[A(X) \to X(\bar{y})] \wedge [B(X)] \ \equiv\ B[X(\bar{t}) := \mathrm{LFP}\, X(\bar{y}).A(X)[\bar{t}]] \qquad (1)$$

– *if B is a first-order formula positive w.r.t. X then*

$$\exists X \forall \bar{y}[X(\bar{y}) \rightarrow A(X)] \wedge [B(X)] \;\equiv\; B[X(\bar{t}) := \textsc{Gfp}\, X(\bar{y}).A(X)[\bar{t}]]. \qquad (2)$$

◁

Remark 4.3. Observe that, whenever formula A in Theorem 4.2 does not contain X, the resulting formula is easily reducible to a first-order formula, as in this case both $\textsc{Lfp}\, X(\bar{y}).A$ and $\textsc{Gfp}\, X(\bar{y}).A$ are equivalent to A. Thus the Ackermann's lemma (see, e.g., [2,15,18]) is subsumed by Theorem 4.2). ◁

An online implementation of the algorithm based on the above theorem is available online (see [13]). Observe that the techniques applied in that algorithm, initiated in [18] and further developed in [6], allow one to transform a large class of formulas to the form required in Theorem 4.2.

Example 4.4. Consider the following second-order formula:

$$\exists X \forall x \forall y[(S(x,y) \vee X(y,x)) \rightarrow X(x,y)] \wedge [\neg X(a,b) \vee \forall z(\neg X(a,z))] \qquad (3)$$

According to Theorem 4.2(1), formula (3) is equivalent to:

$$\neg \textsc{Lfp}\, X(x,y).(S(x,y) \vee X(y,x))[a,b] \vee \forall z(\neg \textsc{Lfp}\, X(x,y).(S(x,y) \vee X(y,x))[a,z]). \qquad (4)$$

Observe that the definition of the least fixpoint appearing in (4) is obtained on the basis of the first conjunct of (3). The successive lines of (4) represent substitutions of $\neg X(a,b)$ and $\forall z(\neg X(a,z))$ of (3) by the obtained definition of the fixpoint. ◁

5 Applications

It can easily be observed that, whenever the elimination of all predicate variables in a formula is possible by applications of Theorem 4.2, the resulting formula is a fixpoint formula over the signature containing equality only. Thus the method applied in the next sections depends on first eliminating all relations appearing in respective formulas and then to apply reasoning in the fixpoint theory of equality.

5.1 Automated Theorem Proving

Introduction. Automated theorem proving in the classical first-order logic is considered fundamental in such applications as formal verification of software[4] and properties of data structures, as well as in the whole spectrum of reasoning techniques appearing in AI, etc. The majority of techniques in these fields are based on various proof systems with resolution-based ones and natural deduction supplemented with algebraic methods, like term rewriting systems etc.

Below we propose another method, which seems to be new in the field. It is not based on any particular proof system. Instead, we first introduce second-order quantifiers in an obvious way, then try to eliminate them and, if this is successful, use the decision procedure for the theory FEQ.

[4] In particular, verification of logic programs, where the method we propose is applicable directly.

The Method. Let $A(R_1, \ldots, R_n)$ be a first-order formula. It is assumed that all relation symbols appearing in this formula are $R_1, \ldots, R_n, =$. In order to prove that $A(R_1, \ldots, R_n)$ is a tautology, $\models A(R_1, \ldots, R_n)$, we prove instead that the following second-order formula

$$\forall R_1 \ldots \forall R_n A(R_1, \ldots, R_n) \tag{5}$$

is a tautology. Of course, $A(R_1, \ldots, R_n)$ is a tautology iff (5) is a tautology. In general this problem is totally undecidable. However, to prove (5) we negate formula (5), eliminate second-order quantifiers $\exists R_1 \ldots \exists R_n$ applying Theorem 4.2 and, if this is successful, apply the decision procedure of Section 3.2. The result is FALSE iff the original formula is equivalent to TRUE.

It should be emphasized that whenever $A(R_1, \ldots, R_n)$ itself is second-order, we can first try to eliminate second-order quantifiers from $A(R_1, \ldots, R_n)$ and then apply the proposed method to the resulting formula. So, in fact, we have a decision procedure for a fragment of the second-order logic, too. This is important in many AI applications, e.g., in reasoning based on various forms of circumscription (see, e.g., [14,11,6]).

Example. Assume a is a constant and consider formula

$$\forall x, y[R(x, y) \rightarrow R(y, x)] \rightarrow [\exists z R(a, z) \rightarrow \exists u R(u, a)] \tag{6}$$

The proof of validity of (6) involves the following steps[5]:

— introduce second-order quantifiers over relations (here only over R):
$\forall R\Big\{\forall x, y[R(x, y) \rightarrow R(y, x)] \rightarrow [\exists z R(a, z) \rightarrow \exists u R(u, a)]\Big\}$
— negate: $\exists R\Big\{\forall x, y[R(x, y) \rightarrow R(y, x)] \wedge \exists z R(a, z) \wedge \forall u \neg R(u, a)\Big\}$
— transform the formula to the form required in Theorem 4.2:
$\exists R\Big\{\forall x, y[R(x, y) \rightarrow (R(y, x) \wedge x \neq a)] \wedge \exists z R(a, z)\Big\}$
— apply Theorem 4.2(2): $\exists z\Big[\text{GFP}\, R(x, y).(R(y, x) \wedge x \neq a)[a, z]\Big]$.

To see that the last formula is FALSE, meaning that the formula (6) is TRUE, we unfold the greatest fixpoint and obtain that

$$\text{GFP}\, R(x, y).(R(y, x) \wedge x \neq a) \;\equiv\; (y \neq a \wedge x \neq a).$$

Thus the resulting formula is equivalent to $\exists z\Big[(y \neq a \wedge x \neq a)[a, z]\Big]$, i.e., to $\exists z\Big[(z \neq a \wedge a \neq a)\Big]$, being equivalent to FALSE. This proves the validity of formula (6).

5.2 Static Verification of Integrity Constraints in Deductive Databases

Introduction. In [9] a method for static verification of integrity constraints in relational databases has been presented. According to the relational database paradigm, integrity constraints express certain conditions that should be preserved by all instances of

[5] These steps can fully be automated, as done in [18,6] and implemented in [13].

a given database, where by an *integrity constraint* we understand a classical first-order formula in the signature of the database. In the existing implementations these conditions are checked dynamically during the database updates. In the case of software systems dealing with rapidly changing environment and reacting in real time, checking integrity constraints after each update is usually unacceptable from the point of view of the required reaction time. Such situations are frequent in many artificial intelligence applications, including autonomous systems.

The Method. In the method of [9] it is assumed that the database can be modified only by well-defined procedures, called *transactions*, supplied by database designers. In such a case the task of verification of integrity constraints reduces to the following two steps:

1. verify that the initial contents of the database satisfies the defined constraints
2. verify that all transactions preserve the constraints.

If both above conditions hold, a simple induction, where the first point is the base step and the second point is the induction step, shows that all possible instances of the database preserve the considered integrity constraints. Of course, the first step can be computed in time polynomial w.r.t. the size of the initial database. In what follows we then concentrate on the second step.

Consider a transaction, which modifies relations $R_1, \ldots, R_n$ giving as a result relations $R'_1, \ldots, R'_n$. The second of the steps mentioned earlier reduces to verification whether the following second-order formula is a tautology:

$$\forall R_1, \ldots, R_n [I(R_1, \ldots, R_n) \rightarrow I(R'_1, \ldots, R'_n)].$$

The method of [9] depends on the application of the Ackermann's lemma of [2], which itself is subsumed by Theorem 4.2 (see Remark 4.3). If the Ackermann's lemma is successful, the resulting formula is expressed in the classical theory of equality, but the requirement is that formulas involved in integrity constraints are, among others, nonrecursive. Therefore [9] considers relational databases rather that deductive ones, which usually require recursion (see, e.g., [1]).

Definition 5.1. *By an* update *of a deductive database DB we shall mean an expression of one of the forms* ADD $\bar{e}$ TO R *or* DELETE $\bar{e}$ FROM R, *where R is an k-ary relation of DB and $\bar{e}$ is a tuple of k elements.* ◁

The meaning of ADD and DELETE updates is rather obvious. Namely, ADD e TO R denotes adding a new tuple e to the relation R, whereas DELETE e FROM R denotes deleting e from R. From the logical point of view, the above updates are formula transformers defined as follows, where $A(R)$ is a formula:

$$\begin{array}{l} (\text{ADD } \bar{e} \text{ TO } R)(A(R(\bar{x}))) \stackrel{\text{def}}{\equiv} A(R(\bar{x}) := (R(\bar{x}) \vee \bar{x} = \bar{e})) \\ (\text{DELETE } \bar{e} \text{ FROM } R(A(R(\bar{x}))) \stackrel{\text{def}}{\equiv} A(R(\bar{x}) := (R(\bar{x}) \wedge \bar{x} \neq \bar{e})). \end{array} \tag{7}$$

Definition 5.2. *By a* transaction *on a deductive database DB we shall mean any finite sequence of* updates *on DB. Transaction T is* correct with respect to integrity constraint $I(R_1, \ldots, R_k)$ *iff the following implication:*

$$I(R_1, \ldots, R_k) \rightarrow T(I(R_1, \ldots, R_k)) \tag{8}$$

is a tautology. ◁

Formula (8) is a tautology iff the following second-order formula is a tautology, too:

$$\forall R_1 \ldots \forall R_k [I(R_1, \ldots, R_k) \rightarrow T(I(R_1, \ldots, R_k))]. \tag{9}$$

In order to eliminate quantifiers $\forall R_1 \ldots \forall R_k$ we first negate (9), as done in Section 5.1:

$$\exists R_1 \ldots \exists R_k [I(R_1, \ldots, R_k) \wedge \neg T(I(R_1, \ldots, R_k))], \tag{10}$$

then try to transform formula (10) into the form suitable for application of Theorem 4.2. This transformation can be based on those given in [6,18] and considered in Section 5.1. If the decision procedure of Section 3.2, applied to (10) results in FALSE, then the formula (9) and, consequently (8), are equivalent to TRUE.

Example. Let $R(x)$ stand for "x is rich", $C(y, x)$ stand for "y is a child of x", j stand for "John" and m for "Mary". Consider the constraint

$$\forall x, y\{[R(x) \wedge C(y, x)] \rightarrow R(y)\} \tag{11}$$

and the transaction ADD $\langle j, m\rangle$ TO C; DELETE $\langle m\rangle$ FROM R.

To prove correctness of the transaction we first consider the formula reflecting (9),

$$\begin{aligned} &\forall C \forall R \Big\{ \forall x, y\{[R(x) \wedge C(y, x)] \rightarrow R(y)\} \rightarrow \\ &\quad \forall x, y\{[R(x) \wedge (C(y, x) \vee (y = j \wedge x = m))] \rightarrow [R(y) \wedge y \neq m]\} \Big\}. \end{aligned} \tag{12}$$

After negating (12) and renaming variables we obtain

$$\begin{aligned} &\exists C \exists R \Big\{ \forall x, y\{[R(x) \wedge C(y, x)] \rightarrow R(y)\} \wedge \\ &\quad \exists u, v\{[R(u) \wedge (C(v, u) \vee (v = j \wedge u = m))] \wedge [\neg R(v) \vee v = m]\} \Big\}. \end{aligned} \tag{13}$$

Some transformations of (13) made in the spirit of algorithms [18,6,13] result in

$$\begin{aligned} &\exists u, v \exists C \exists R \Big\{ \forall x, y\{[R(x) \wedge C(y, x)] \rightarrow R(y)\} \wedge [\neg R(v) \vee v = m] \wedge \\ &\quad R(u) \wedge \forall x, y[(x = v \wedge y = u \wedge (v \neq j \vee u \neq m)) \rightarrow C(x, y)] \Big\}. \end{aligned}$$

We first eliminate $\exists C$ which, according to Remark 4.3, results in the following formula without fixpoints

$$\begin{aligned} &\exists u, v \exists R \Big\{ \forall x, y\{[R(x) \wedge y = v \wedge x = u \wedge (v \neq j \vee u \neq m)] \rightarrow R(y)\} \wedge \\ &\qquad [\neg R(v) \vee v = m] \wedge R(u) \Big\}, \end{aligned}$$

equivalent to

$$\exists u, v \exists R \Big\{ \\ \forall y \{ [\exists x [R(x) \wedge y = v \wedge x = u \wedge (v \neq j \vee u \neq m)] \vee y = u] \rightarrow R(y) \} \wedge \\ [\neg R(v) \vee v = m] \Big\}.$$

Now an application of Theorem 4.2(1) results in

$$\exists u, v \Big\{ v = m \vee \\ \neg \textsc{Lfp}\, R(y).[\exists x [R(x) \wedge y = v \wedge x = u \wedge (v \neq j \vee u \neq m)] \vee y = u] \Big\}. \tag{14}$$

Applying the decision procedure of Section 3.2 shows that formula (14) is equivalent to FALSE, which proves correctness of the considered transaction.

5.3 Reasoning with Weakest Sufficient and Strongest Necessary Conditions

Introduction. Weakest sufficient and strongest necessary conditions have been introduced by Lin in [12] in the context of propositional reasoning and extended to the first-order case in [7].

Consider a formula A expressed in some logical language. Assume that one is interested in approximating A in a less expressive language, say L, which allows for more efficient reasoning. A sufficient condition of A, expressed in L, is a formula implying A and a necessary condition of A, expressed in L, is a formula implied A. Thus the weakest sufficient condition provides "the best" approximation of A that guarantees its satisfiability and the strongest necessary condition provides "the best" approximation of A that still cannot exclude A, both expressed in the less expressive language.

Let us emphasize that sufficient and necessary conditions are vital for providing solutions to important problems concerning, e.g., approximate reasoning, abduction and hypotheses generation, building communication interfaces between agents or knowledge compilation.

Below we assume that theories are finite, i.e., can be expressed by finite conjunctions of axioms.

The Method. The following are definitions for necessary and sufficient conditions of a formula A relativized to a subset $\bar{P}$ of relation symbols under a theory T, as introduced in [12].

Definition 5.3. *By a* necessary condition of a formula A on the set of relation symbols $\bar{P}$ under theory T *we shall understand any formula B containing only symbols in $\bar{P}$ such that $T \models A \rightarrow B$. It is the* strongest necessary condition*, denoted by* $\textsc{Snc}\left(A; T; \bar{P}\right)$ *if, additionally, for any necessary condition C of A on $\bar{P}$ under T, we have $T \models B \rightarrow C$.*

By a sufficient condition of a formula A on the set of relation symbols $\bar{P}$ under theory T *we shall understand any formula B containing only symbols in $\bar{P}$ such that $T \models B \rightarrow A$. It is the* weakest sufficient condition*, denoted by* $\textsc{Wsc}\left(A; T; \bar{P}\right)$ *if, additionally, for any sufficient condition C of A on $\bar{P}$ under T, we have $T \models C \rightarrow B$.*$\triangleleft$

The set $\bar{P}$ in Definition 5.3 is referred to as the *target language*.

According to [7], we have the following characterization of weakest sufficient and strongest necessary conditions.

Lemma 5.4. *For any formula A, any set of relation symbols $\bar{P}$ and a closed theory T:*

1. SNC $\left(A; T; \bar{P}\right)$ *is defined by* $\exists \bar{X}[T \wedge A]$
2. WSC $\left(A; T; \bar{P}\right)$ *is defined by* $\forall \bar{X}[T \rightarrow A]$,

where $\bar{X}$ consists of all relation symbols appearing in T or A, but not in $\bar{P}$. ◁

Thus, reasoning with weakest sufficient and strongest necessary conditions can again, in many cases, be reduced to FEQ. Namely, one can first try to eliminate second-order quantifiers from second-order formulas appearing in characterizations provided in Lemma 5.4 and then to apply the method of Section 5.1.

The method is best visible in the case when we are interested in formulas of the target language implied by SNC $\left(A; T; \bar{P}\right)$ and implying WSC $\left(A; T; \bar{P}\right)$. In these cases we deal with formulas of the form $\forall \bar{R}\{\exists \bar{X}[T \wedge A] \rightarrow B\}$ and $\forall \bar{R}\{B \rightarrow \forall \bar{X}[T \rightarrow A]\}$, where $\bar{R}$ consists of all relation symbols appearing free in the respective formulas and B contains no relation symbols of $\bar{X}$. Of course, these forms are equivalent to $\forall \bar{R} \forall \bar{X}\{[T \wedge A] \rightarrow B\}$ and $\forall \bar{R} \forall \bar{X}\{B \rightarrow [T \rightarrow A]\}$. Thus the proposed method applies here in an obvious manner.

Example. Consider a theory given by formula (11) of Section 5.2 and

$$\text{SNC}\left(\neg R(m) \wedge R(j); (11); \{C\}\right). \tag{15}$$

Suppose we are interested in verifying whether (15) implies $\exists x, y \neg C(y, x)$.

According to Lemma 5.4, (15) is equivalent to

$$\exists R \forall x, y\{[R(x) \wedge C(y, x)] \rightarrow R(y)\} \wedge \neg R(m) \wedge R(j).$$

i.e., we are interested in verifying whether

$$\forall R \forall C\{\forall x, y\{[R(x) \wedge C(y, x)] \rightarrow R(y)\} \wedge \neg R(m) \wedge R(j)\} \rightarrow \exists x, y \neg C(y, x), \tag{16}$$

i.e., whether

$$\neg \exists R \exists C \forall x, y\{[C(y, x) \rightarrow [R(x) \rightarrow R(y)]\} \wedge \neg R(m) \wedge R(j) \wedge \forall x, y C(y, x). \tag{17}$$

According to Theorem 4.2(2), the elimination of $\exists C$ from (17) results in

$$\neg \exists R \forall x, y[R(x) \rightarrow R(y)] \wedge \neg R(m) \wedge R(j),$$

which is equivalent to

$$\neg \exists R \forall y[(y = j \vee \exists x R(x)) \rightarrow R(y)] \wedge \neg R(m). \tag{18}$$

According to Theorem 4.2(1), the elimination of $\exists R$ from (18) results in

$$\neg\neg \text{LFP}\, R(y).[y = j \vee \exists x R(x)][m].$$

Applying the decision procedure of Section 3.2, one can easily verify that the above formula is TRUE only when $m = j$.

6 Conclusions

In the current paper we have investigated the fixpoint theory of equality and have shown its applications in automatizing various forms of reasoning in theorem proving, deductive databases and artificial intelligence.

Since many non-classical logics can be translated into the classical logic, the method is applicable to non-classical logics, too.

References

1. S. Abiteboul, R. Hull, and V. Vianu. *Foundations of Databases*. Addison-Wesley Pub. Co., 1996.
2. W. Ackermann. Untersuchungen über das eliminationsproblem der mathematischen logik. *Mathematische Annalen*, 110:390–413, 1935.
3. J. Barwise. On Moschovakis closure ordinals. *Journal of Symbolic Logic*, 42:292–296, 1977.
4. A. Chandra and D. Harel. Computable queries for relational databases. *Journal of Computer and System Sciences*, 21:156–178, 1980.
5. A. Chandra and D. Harel. Structure and complexity of relational queries. *Journal of Computer and System Sciences*, 25:99–128, 1982.
6. P. Doherty, W. Łukaszewicz, and A. Szałas. Computing circumscription revisited. *Journal of Automated Reasoning*, 18(3):297–336, 1997.
7. P. Doherty, W. Łukaszewicz, and A. Szałas. Computing strongest necessary and weakest sufficient conditions of first-order formulas. *International Joint Conference on AI (IJCAI'2001)*, pages 145 – 151, 2000.
8. N. Immerman. Upper and lower bounds for first-order expressibility. *Journal of Computer and System Sciences*, 25:76–98, 1982.
9. J. Kachniarz and A. Szałas. On a static approach to verification of integrity constraints in relational databases. In E. Orłowska and A. Szałas, editors, *Relational Methods for Computer Science Applications*, pages 97–109. Springer Physica-Verlag, 2001.
10. Phokion Kolaitis and Moshe Vardi. Ph. kolaitis and m. vardi. on the expressive power of variable-confined logics. In *Proc. IEEE Conf. Logic in Computer Science*, pages 348–359, 1996.
11. V. Lifschitz. Circumscription. In D. M. Gabbay, C. J. Hogger, and J. A. Robinson, editors, *Handbook of Artificial Intelligence and Logic Programming*, volume 3, pages 297–352. Oxford University Press, 1991.
12. F. Lin. On strongest necessary and weakest sufficient conditions. In A.G. Cohn, F. Giunchiglia, and B. Selman, editors, *Proc. 7th International Conf. on Principles of Knowledge Representation and Reasoning, KR2000*, pages 167–175. Morgan Kaufmann Pub., Inc., 2000.
13. M. Magnusson. DLS*. *http://www.ida.liu.se/labs/kplab/projects/dlsstar/*, 2005.
14. J. McCarthy. Circumscription: A form of non-monotonic reasoning. *Artificial Intelligence Journal*, 13:27–39, 1980.
15. A. Nonnengart, H.J. Ohlbach, and A. Szałas. Elimination of predicate quantifiers. In H.J. Ohlbach and U. Reyle, editors, *Logic, Language and Reasoning. Essays in Honor of Dov Gabbay, Part I*, pages 159–181. Kluwer, 1999.

16. A. Nonnengart and A. Szałas. A fixpoint approach to second-order quantifier elimination with applications to correspondence theory. In E. Orłowska, editor, *Logic at Work: Essays Dedicated to the Memory of Helena Rasiowa*, volume 24 of *Studies in Fuzziness and Soft Computing*, pages 307–328. Springer Physica-Verlag, 1998.
17. B. Poizat. Deux ou trois choses que je sais de L_n. *Journal of Symbolic Logic*, 47:641–658, 1982.
18. A. Szałas. On the correspondence between modal and classical logic: An automated approach. *Journal of Logic and Computation*, 3:605–620, 1993.