Proving Semantic Properties as First-Order Satisfiability☆

Salvador Lucas[1]

[a] *Valencian Research Institute for Artificial Intelligence (VRAIN)*
Universitat Politècnica de València, Spain
http://slucas.webs.upv.es/

Abstract

The semantics of computational systems (e.g., relational and knowledge data bases, query-answering systems, programming languages, etc.) can often be expressed as (the specification of) a logical theory Th. Queries, goals, and claims about the behavior or features of the system can be expressed as formulas φ which should be checked with respect to the *intended model* of Th, which is often huge or even incomputable. In this paper we show how to prove such *semantic* properties φ of Th by just *finding a model* $\mathcal{A}$ of $\mathsf{Th} \cup \{\varphi\} \cup \mathcal{Z}_\varphi$, where $\mathcal{Z}_\varphi$ is an appropriate (possibly empty) theory depending on φ only. Applications to relational and deductive databases, rewriting-based systems, logic programming, and answer set programming are discussed.

Keywords: Automated reasoning, First-Order Logic, Logical models

1. Introduction

Artificial Intelligence (AI) is a main 'user' of logic (see [61] for a recent survey). In his 1995 book [45], Robert C. Moore distinguishes three main roles of logic in AI: as an *analytical tool*, as a *knowledge representation and reasoning system*, and as a *programming language*. For instance, knowledge databases can be represented as theories of more or less sophisticated logics, like in relational [10], or deductive databases [48]. Queries are presented as (or translated into) specific formulas φ to be proved with respect to the underlying theory [22]. In a programming setting, declarative programming languages are intended to represent programs as (the specification of) theories of a given logic and the execution of the program as deduction of a goal φ representing some computation. Logic, functional, and rewriting-based programming languages (among others) can be seen in this way. Imperative programs and their operational semantics can also be represented as a theory of a given logic and its execution

☆Partially supported by the EU (FEDER), and projects RTI2018-094403-B-C32, PROMETEO/2019/098, and SP20180225.

investigated by using logic [38]. Besides the intended or 'normal' deductive use of logic to implement the expected functionality of databases (queries, checking functional dependencies and integrity constraints, etc.) or programs (executing them), logic can also be used to discover *properties* of the systems, by means of formulas φ involving symbols which are meaningful for the user. From a logical point of view, in all these cases the following approach is naturally adopted [22]:

$$\text{can } \varphi \text{ be } \textit{proved} \text{ from } \mathsf{Th}? \text{ (written } \mathsf{Th} \vdash \varphi) \tag{1}$$

Example 1. *Consider the following relational database about teachers* $(\mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{d})$ *and students* $(\mathtt{p}, \mathtt{q}, \mathtt{r})$ *together with a relation* `teach` *giving information about the students taught by each teacher, according to the* table *[50, page 59]*

`teach`	*Teacher*	*Student*
	`a`	`p`
	`a`	`q`
	`b`	`q`
	`c`	`r`

This table describes a (many-sorted) first-order theory Teach *consisting of the following* facts*:*

$$\mathtt{teach}(\mathtt{a}, \mathtt{p}), \quad \mathtt{teach}(\mathtt{b}, \mathtt{q}), \quad \mathtt{teach}(\mathtt{a}, \mathtt{q}), \quad \mathtt{teach}(\mathtt{c}, \mathtt{r})$$

where `teach` *is a binary predicate with arguments of sort Teacher and Student, respectively. The question "*is `a` teaching someone?*" would be encoded as:*

$$(\exists y : \textit{Student})\, \mathtt{teach}(\mathtt{a}, y) \tag{2}$$

McCune's (first-order) theorem prover Prover9 *[39] obtains a proof of* (2)*.*

However, in some cases, the approach (1) may lead to counterintuitive results.

Example 2. *When faced to the question "*is `d` teaching someone?*", i.e.,*

$$(\exists y : \textit{Student})\, \mathtt{teach}(\mathtt{d}, y) \tag{3}$$

Prover9 *fails to obtain a proof of* (3)*. An attempt to prove the negation of (3),*

$$\neg(\exists y : \textit{Student})\, \mathtt{teach}(\mathtt{d}, y) \tag{4}$$

with Prover9 *also* fails*. Now, following* Prover9/Mace4 *documentation, "*if the statement is the denial of some conjecture, any structures[1] found by Mace4 are counterexamples to the conjecture*".*[2] *With* Mace4 *we obtain a* countermodel[3]

[1] A structure is an *interpretation* of the function and predicate symbols in the usual sense, see Section 2 below.

[2] `https://www.cs.unm.edu/~mccune/prover9/manual/2009-11A/mace4.html`

[3] i.e., a model of the theory together with the *negation* of the considered formula.

*of (3). We would then conclude that (4) holds. Consider now the claim "*every student is taught by some teacher*", i.e.,*

$$(\forall y : \mathit{Student})(\exists x : \mathit{Teacher})\, \mathtt{teach}(x, y) \tag{5}$$

Prover9 *fails to prove this goal, but* Mace4 *obtains a countermodel. However, we expect (5) to hold; what should we think of such a 'countermodel'?*

In order to clarify this issue and answer the question posed in the last sentence of Example 2, first note that the provability-based information retrieval/analysis approach represented by (1) is equivalent, when using a sound and complete proof method,[4] to

$$\text{is } \varphi \text{ a } \textit{logical consequence} \text{ of } \mathsf{Th}? \text{ (written } \mathsf{Th} \models \varphi) \tag{6}$$

where $\mathsf{Th} \models \varphi$ means that φ holds in *all* models of Th, i.e., each interpretation $\mathcal{A}$ that makes true the formulas in Th, also makes φ true.

1.1. Our contribution

As remarked by Clark, sentences expressing properties should be checked *with respect to a given canonical model only* [7, Chapter 4]. Similarly, in the logical approach to relational databases [46], solving queries and checking functional dependencies and integrity constraints is thought as the *evaluation* of logical formulas φ with respect to the facts stored in the database which are considered as a logical *interpretation* (or model). This perspective makes a difference when trying to check whether a computational system has a property or not by using logic techniques and tools.

Example 3. *Note that there are models of* Teach *which do not satisfy* (3) *(e.g., the interpretation which makes true the atoms in* Teach *only); and there are models of* Teach *which do not satisfy* $\neg(3)$ *(e.g., the interpretation which makes* all *atoms* $\mathtt{teach}(t, s)$ *for all teachers* t *and students* s *true). Thus, neither* (3) *nor* $\neg(3)$ *are logical consequences of* Teach *and (by the standard correspondence between provability and logical consequence) they are* not *provable in* Teach*. Still, we expect* $\neg(3)$ *to hold in* Teach*.*

Relational databases (viewed as a set of ground atoms representing related tuples of values), logic programs consisting of Horn clauses only [14], rewriting logic programs [42] (viewed as first-order theories), etc., have a canonical model $\mathcal{I}_{\mathsf{Th}}$ which is the set of atoms (without variables) which can be deduced from the corresponding theory. Thus, we naturally consider the following:

$$\text{is } \varphi \text{ satisfiable in } \mathcal{I}_{\mathsf{Th}}? \text{ (written } \mathcal{I}_{\mathsf{Th}} \models \varphi) \tag{7}$$

[4] resolution, Hilbert's axiomatic calculus, natural deduction, etc.

Remark 4 (Theorems vs. properties). *Note that φ in (7) does* not *need to be a theorem of* Th *(as in (1)); thus, we better generically call φ in (7) a* semantic property *of* Th *(precise definitions are given in Section 3).*

In this paper, we investigate an alternative method to verify semantic properties formulated as in (7). We exploit the *preservation* of formulas under homomorphisms h between interpretations. A homomorphism h *preserves* a formula φ if φ is satisfied in the target interpretation of h whenever φ is satisfied in its domain interpretation [25, Section 2.4]. Hence, if the interpretation homomorphism h from $\mathcal{I}_{\mathsf{Th}}$ to a given model $\mathcal{A}$ of Th preserves φ, then proving the *negation* of φ satisfiable in $\mathcal{A}$ implies that $\neg\varphi$ holds in $\mathcal{I}_{\mathsf{Th}}$, i.e., $\neg\varphi$ is a semantic property of Th. Homomorphisms preserve *Existentially Closed Boolean Combinations of Atoms (ECBCA)*, i.e., formulas $(\exists \vec{x}) \bigvee_i \bigwedge_j A_{ij}$, where A_{ij} are atoms.

Example 5. *(cont. Example 2) Since* (3) *is an ECBCA, the model of* Teach *and* $\neg(3)$ *obtained by* $\mathsf{Mace4}$ *(with domain* $\{0,1\}$*, interpreting* `d` *as* 1 *and any other constant symbol as* 0*, and where only* `teach(0,0)` *holds) faithfully proves that property* $\neg(3)$ *holds in* Teach *(see Example 36 below for technical details).*

Homomorphisms preserve other first-order sentences if further requirements are imposed: (i) *positive* sentences (where connective '$\neg$' is absent) are preserved under *surjective* homomorphisms[5] and (ii) *arbitrary* sentences are preserved under *embeddings* [25, Theorem 2.4.3].

Example 6. *(cont. Example 2) Note that (5) is not an ECBCA due to the universal quantification. The model of* Teach *and* $\neg(5)$ *obtained by* $\mathsf{Mace4}$ *has domain* $\{0,1\}$*, all constant symbols are interpreted as 0, and only* `teach(0,0)` *holds. Note that the interpretation homomorphism is* not *surjective as the domain value* 1 *corresponds to no constant symbol in the signature. Thus, we cannot conclude that property* $\neg(5)$ *holds in* Teach*.*

Our technique is summarized as follows: given a theory Th and a sentence φ, we show that $\mathcal{I}_{\mathsf{Th}} \models \varphi$ holds if there is an interpretation $\mathcal{A}$ satisfying

$$\mathcal{A} \models \mathsf{Th} \cup \mathsf{H} \cup \mathsf{N} \cup \{\varphi\} \tag{8}$$

for appropriate theories H and N which only depend on φ and can be empty. Theory H guarantees surjectivity of the homomorphism (when required). Theory N deals with the presence of negative literals (if any).

1.2. Structure of the paper

After some preliminaries in Section 2, Section 3 explains our notion of semantic property of a first-order theory. Section 4 summarizes a number of relevant fields where our techniques could be applied: relational and deductive

[5] A mapping $f : A \to B$ is *surjective* if for all $b \in B$ there is $a \in A$ such that $f(a) = b$.

databases, logic programming, answer set programming, and rewriting-based systems. We introduce some running examples further developed along the paper. In Section 5 we provide a preservation theorem that improves [25]. In contrast to [25], we focus on *many-sorted logic* [63] (see Section 2). This has an advantage: since homomorphisms in many-sorted logic with set of sorts S are actually a *family* h_s of homomorphisms between components of sort s for each $s \in S$, the preservation requirements for h_s depend on the *specific quantification* of variables $x : s$ *for such a sort*. Section 6 investigates how to guarantee surjectivity of homomorphisms. Section 7 discusses the possibility of providing more information about *disproved* properties by means of *refutation witnesses*, i.e., (counter)examples of *sentences* which are synthesized from the models that are used to disprove the property. Section 8 enumerates some explored applications of our techniques. Section 9 discusses some related work. Section 10 concludes.

This paper is an extended and completely revised version of [34].[6] In particular, Sections 3, 4, and 5.2 are completely new. Also, whilst examples in [34] centered the attention in rewriting-based systems, in this paper we show the generality of our approach by also considering examples from relational and deductive databases, logic programming, etc.

2. Many-Sorted First-Order Logic

Given a set of *sorts* S, a *(many-sorted) signature (with predicates)* $\Omega = (S, \Sigma, \Pi)$ consists of a set of sorts S, an $S^* \times S$-indexed family of sets $\Sigma = \{\Sigma_{w,s}\}_{(w,s)\in S^*\times S}$ containing *function symbols* $f \in \Sigma_{s_1\cdots s_k,s}$, with a rank declaration $f : s_1 \cdots s_k \to s$ (constant symbols c have rank declaration $c : \lambda \to s$, where λ denotes the *empty* sequence), and an S^+-indexed family of sets $\Pi = \{\Pi_w\}_{w\in S^+}$ of ranked predicates $P : w$. Given an S-sorted set $\mathcal{X} = \{\mathcal{X}_s \mid s \in S\}$ of *mutually disjoint* sets of variables (which are also disjoint from Σ), the set $\mathcal{T}_\Sigma(\mathcal{X})_s$ of terms of sort s is the least set such that $\mathcal{X}_s \subseteq \mathcal{T}_\Sigma(\mathcal{X})_s$ and for each $f : s_1 \dots s_k \to s$ and $t_i \in \mathcal{T}_\Sigma(\mathcal{X})_{s_i}$, $1 \le i \le k$, $f(t_1, \dots, t_k) \in \mathcal{T}_\Sigma(\mathcal{X})_s$. If $\mathcal{X} = \emptyset$, we write $\mathcal{T}_\Sigma$ rather than $\mathcal{T}_\Sigma(\emptyset)$ for the set of *ground* terms. The set $\mathcal{T}_\Sigma(\mathcal{X})$ of *many-sorted terms* is $\mathcal{T}_\Sigma(\mathcal{X}) = \bigcup_{s\in S} \mathcal{T}_\Sigma(\mathcal{X})_s$. For $w = s_1 \cdots s_n \in S^+$, we write $\mathcal{T}_\Sigma(\mathcal{X})_w$ rather than $\mathcal{T}_\Sigma(\mathcal{X})_{s_1} \times \cdots \times \mathcal{T}_\Sigma(\mathcal{X})_{s_n}$ and also $\vec{t} \in \mathcal{T}_\Sigma(\mathcal{X})_w$ rather than $t_i \in \mathcal{T}_\Sigma(\mathcal{X})_{s_i}$ for each $1 \le i \le n$. The formulas $\varphi \in \mathit{Form}_\Omega$ of a signature Ω are built up from atoms $P(\vec{t})$ with $P \in \Pi_w$ and $\vec{t} \in \mathcal{T}_\Sigma(\mathcal{X})_w$, logic connectives ($\neg$, $\wedge$, and also $\vee$, $\Rightarrow$,...) and quantifiers ($\forall$ and $\exists$) in the usual way. A closed formula, i.e., one whose variables are all universally or existentially quantified, is called a *sentence*. A theory is a set of sentences (from a signature Ω (sometimes we call it an Ω-theory). Substitutions σ are S-sorted mappings such that for all sorts $s \in S$, we have $\sigma(x) \in \mathcal{T}_\Sigma(\mathcal{X})_s$.

An Ω-*structure* $\mathcal{A}$ consists of (i) an S-sorted family $\{\mathcal{A}_s \mid s \in S\}$ of sets called the *carriers* or *domains* together with (ii) a function $f^{\mathcal{A}}_{w,s} \in \mathcal{A}_w \to \mathcal{A}_s$ for each

[6] LOPSTR 2018 best paper award; it also continues and subsumes the results in [33].

$f \in \Sigma_{w,s}$ ($\mathcal{A}_w$ is a one point set when $w = \lambda$ and hence $\mathcal{A}_w \to \mathcal{A}_s$ is isomorphic to $\mathcal{A}_s$), and (iii) an assignment to each $P \in \Pi_w$ of a subset $P_w^{\mathcal{A}} \subseteq \mathcal{A}_w$; if the identity predicate $\_ = \_ : ss$ is in Π_{ss}, then $(=)_{s\,s}^{\mathcal{A}} = \{(a, a) \mid a \in \mathcal{A}_s\}$, i.e., $\_ = \_ : ss$ is interpreted as the identity on $\mathcal{A}_s$.

A *Herbrand interpretation* $\mathcal{H}$ is an Ω-structure with S-sorted *Herbrand Domains* $\mathcal{T}_{\Sigma s}$ of ground terms for $s \in S$. Given $f : w \to s$ and $\vec{t} \in \mathcal{T}_{\Sigma w}$, we let $f_{w,s}^{\mathcal{H}}(\vec{t}) = f(\vec{t})$. Given $P : w$, $P_w^{\mathcal{H}}$ is a subset of $\mathcal{T}_{\Sigma w}$. Since the interpretation of domains and function symbols is fixed, $\mathcal{H}$ is often just presented as a set of atoms $\bigcup_{w \in S^+} \bigcup_{P \in \Pi_w} \{P(\vec{t}) \mid \vec{t} \in P_w^{\mathcal{H}}\}$. Viceversa: every set of atoms Th yields a Herbrand interpretation $\mathcal{H}_\Omega(\mathsf{Th})$ (or just $\mathcal{H}(\mathsf{Th})$ if no confusion arises).

Let $\mathcal{A}$ and $\mathcal{A}'$ be Ω-structures. An *Ω-homomorphism* $h : \mathcal{A} \to \mathcal{A}'$ is an S-sorted function $h = \{h_s : \mathcal{A}_s \to \mathcal{A}'_s \mid s \in S\}$ such that for each $f \in \Sigma_{w,s}$ and $P \in \Pi_w$ with $w = s_1, \ldots, s_k$, (i) $h_s(f_{w,s}^{\mathcal{A}}(a_1, \ldots, a_k)) = f_{w,s}^{\mathcal{A}'}(h_{s_1}(a_1), \ldots, h_{s_k}(a_k))$ and (ii) if $\vec{a} \in P_w^{\mathcal{A}}$, then $h(\vec{a}) \in P_w^{\mathcal{A}'}$. Given an S-sorted *valuation mapping* $\alpha : \mathcal{X} \to \mathcal{A}$, the evaluation mapping $[\_]_\alpha^{\mathcal{A}} : \mathcal{T}_\Sigma(\mathcal{X}) \to \mathcal{A}$ is the unique (S, Σ)-homomorphism extending α. Finally, $[\_]_\alpha^{\mathcal{A}} : \mathit{Form}_\Omega \to \mathit{Bool}$ is given by:

1. $[P(t_1, \ldots, t_n)]_\alpha^{\mathcal{A}} = \mathit{true}$ (with $P \in \Pi_w$) iff $([t_1]_\alpha^{\mathcal{A}}, \ldots, [t_n]_\alpha^{\mathcal{A}}) \in P_w^{\mathcal{A}}$;
2. $[\neg\varphi]_\alpha^{\mathcal{A}} = \mathit{true}$ iff $[\varphi]_\alpha^{\mathcal{A}} = \mathit{false}$;
3. $[\varphi \wedge \psi]_\alpha^{\mathcal{A}} = \mathit{true}$ iff $[\varphi]_\alpha^{\mathcal{A}} = \mathit{true}$ and $[\psi]_\alpha^{\mathcal{A}} = \mathit{true}$; and
4. $[(\forall x : s)\, \varphi]_\alpha^{\mathcal{A}} = \mathit{true}$ iff for all $a \in \mathcal{A}_s$, $[\varphi]_{\alpha[x \mapsto a]}^{\mathcal{A}} = \mathit{true}$.

The interpretation of ground terms t does not depend on any valuation α; thus, we write $[t]^{\mathcal{A}}$ or even $t^{\mathcal{A}}$ to refer its value. The truth value of sentences φ does not depend on any valuation α; we write $[\varphi]^{\mathcal{A}}$ or $\varphi^{\mathcal{A}}$ to denote it.

A valuation $\alpha \in \mathcal{X} \to \mathcal{A}$ *satisfies* φ in $\mathcal{A}$ (written $\mathcal{A} \models \varphi\,[\alpha]$) if $[\varphi]_\alpha^{\mathcal{A}} = \mathit{true}$. We then say that φ is *satisfiable*. If $\mathcal{A} \models \varphi\,[\alpha]$ for *all* valuations α, we write $\mathcal{A} \models \varphi$ and say that $\mathcal{A}$ is a *model* of φ or that φ is *true* in $\mathcal{A}$. We say that $\mathcal{A}$ is *a model of a set of sentences* $\mathsf{Th} \subseteq \mathit{Form}_\Omega$ (written $\mathcal{A} \models \mathsf{Th}$) if for all $\varphi \in \mathsf{Th}$, $\mathcal{A} \models \varphi$. Given a sentence φ, we write $\mathsf{Th} \models \varphi$ (φ is a logical consequence of Th) iff $\mathcal{A} \models \varphi$ holds for *all models* $\mathcal{A}$ of Th. Let $\mathit{Mod}(\mathsf{Th})$ be the class of structures $\mathcal{A}$ which are models of Th. Two sets of sentences Th and Th' are *logically equivalent* if $\mathit{Mod}(\mathsf{Th}) = \mathit{Mod}(\mathsf{Th}')$. If φ can be *proved* from Th by using an appropriate (sound and complete) calculus (see footnote 4) we write $\mathsf{Th} \vdash \varphi$.

A *literal* is an atom or the negation of an atom. A *clause* is a disjunction of literals. A *set of clauses* $\mathcal{C}$ is regarded as a conjunction of all clauses in $\mathcal{C}$, where every variable in $\mathcal{C}$ is *universally quantified* [5]. A theory all whose sentences are clauses is often called a *clausal theory*. For every sentence $\varphi \in \mathit{Form}_\Omega$ there is a sentence φ' in *clausal form* which is inconsistent iff φ is [5, Section 4.2] (see [55] for the MS-FOL setting). A *Horn clause* is a clause $\neg A_1 \vee \cdots \vee \neg A_n \vee B$ with at most one non-negated atom.

3. Semantic Properties of First-Order Theories

Given a signature Ω, consider the the set of ground atoms which can be proved from Th:

$$\mathsf{Th}^{\vdash} \quad = \quad \{A \mid \mathsf{Th} \vdash A, \text{where } A \text{ is a ground atom}\} \tag{9}$$

$\mathsf{Th}^{\vdash}$ can be seen as a Herbrand interpretation $\mathcal{I}_{\mathsf{Th}} = \mathcal{H}_{\Omega}(\mathsf{Th}^{\vdash})$. For every set Th of ground atoms, we have $\mathsf{Th} = \mathsf{Th}^{\vdash}$. For arbitrary theories Th we have the following.

Proposition 7. *Every model of a theory* Th *is a model of* $\mathsf{Th}^{\vdash}$.

Proof. For all ground atoms A,

$$\begin{array}{rcll} A \in \mathsf{Th}^{\vdash} & \Leftrightarrow & \mathsf{Th} \vdash A & \text{by (9)} \\ & \Leftrightarrow & \mathsf{Th} \models A & \text{sound \& complete} \\ & \Leftrightarrow & (\forall \mathcal{A}) \mathcal{A} \models \mathsf{Th} \Rightarrow \mathcal{A} \models A & \text{logical consequence} \end{array}$$

If $\mathcal{A} \models \mathsf{Th}$ holds for some structure $\mathcal{A}$, then $\mathcal{A} \models A$ holds and $\mathcal{A} \models \mathsf{Th}^{\vdash}$. □

Thus, $Mod(\mathsf{Th}) \subseteq Mod(\mathsf{Th}^{\vdash})$. van Emden and Kowalski [14] proved that, for theories Th consisting of clauses, $\mathsf{Th}^{\vdash}$ is the intersection of all Herbrand models of Th (and hence $\mathsf{Th}^{\vdash} \subseteq \mathcal{M}$ for all Herbrand model $\mathcal{M}$ of Th). However, $\mathcal{I}_{\mathsf{Th}}$ may fail to be a model.

Example 8. *[14] For* $\mathsf{Th} = \{p(a) \vee p(b)\}$*, with* a *and* b *constants, we have* $\mathsf{Th}^{\vdash} = \emptyset$*. Although* $\mathcal{H} = \{p(a)\}$ *and* $\mathcal{H}' = \{p(b)\}$ *are (minimal) models of* Th*,* $\mathcal{I}_{\mathsf{Th}} \not\models \mathsf{Th}$.

Thus, in general $Mod(\mathsf{Th}) \not\supseteq Mod(\mathsf{Th}^{\vdash})$. Furthermore, we have the following.

Proposition 9. *If a clausal theory* Th *has a least Herbrand model, it is* $\mathcal{I}_{\mathsf{Th}}$.

Proof. Let $\mathcal{M}$ be a least Herbrand model of Th different from $\mathcal{I}_{\mathsf{Th}}$. Since $\mathsf{Th}^{\vdash} \subseteq \mathcal{M}$, there is $A \in \mathcal{M} - \mathsf{Th}^{\vdash}$. Since $\mathcal{M}$ is the least Herbrand model of Th, for every Herbrand model $\mathcal{M}'$ of Th, we have $A \in \mathcal{M} \subseteq \mathcal{M}'$. Thus, A is in the intersection of all Herbrand models of Th. Since $\mathsf{Th}^{\vdash}$ is the intersection of all Herbrand models of Th, $A \in \mathsf{Th}^{\vdash}$ leading to a contradiction. □

In the following, given an Ω-structure $\mathcal{A}$, $h_{\mathcal{A}} : \mathcal{I}_{\mathsf{Th}} \to \mathcal{A}$ (or just h) is the *interpretation mapping* $h_{\mathcal{A}}(t) = t^{\mathcal{A}}$ that associates $t^{\mathcal{A}}$ to each ground term t.

Proposition 10. *For each model* $\mathcal{A}$ *of a theory* Th*,* $h_{\mathcal{A}}$ *is the unique homomorphism from* $\mathcal{I}_{\mathsf{Th}}$ *to* $\mathcal{A}$.

Proof. By Proposition 7, $\mathcal{A}$ models $\mathsf{Th}^{\vdash}$, which is a set of ground atoms. Thus, the theorem follows by [25, Theorem 1.5.2]; although Hodges' result concerns unsorted FOL, it is also valid in our many-sorted setting ([20, Section 3.2]). □

Definition 11 (Semantic property). *Let* Th *be a theory and* $\varphi \in Form(\mathsf{Th})$. *We say that* φ *is an* Th-property *if* $\mathcal{I}_{\mathsf{Th}} \models \varphi$ *holds; if* $\mathcal{I}_{\mathsf{Th}} \models \mathsf{Th}$ *also holds, we say that* φ *is a* semantic property *of* Th.

Propositions 9 and 10 motivate our choice of $\mathcal{I}_{\mathsf{Th}}$ as a semantic reference for a theory Th. Our purpose is using arbitrary models $\mathcal{A}$ of theories Th to prove properties of such theories given as first-order formulas φ. The existence of a unique homomorphism from $\mathcal{I}_{\mathsf{Th}}$ to any model $\mathcal{A}$ is essential to use the preservation results in Section 5. As mentioned in Remark 4, we have the following.

Proposition 12. *Every logical consequence* φ *of a theory* Th *such that* $\mathcal{I}_{\mathsf{Th}} \models \mathsf{Th}$ *holds is a semantic property of* Th*:* $\mathsf{Th} \vdash \varphi$ *implies* $\mathcal{I}_{\mathsf{Th}} \models \mathsf{Th} \cup \{\varphi\}$.

Proof. $\mathsf{Th} \vdash \varphi$ is equivalent to $\mathsf{Th} \models \varphi$. Thus, for all structures $\mathcal{A}$, $\mathcal{A} \models \mathsf{Th}$ implies $\mathcal{A} \models \varphi$. If $\mathcal{I}_{\mathsf{Th}} \models \mathsf{Th}$ holds, then $\mathcal{I}_{\mathsf{Th}} \models \varphi$ holds, i.e., $\mathcal{I}_{\mathsf{Th}} \models \mathsf{Th} \cup \{\varphi\}$. □

Thus, theorem proving can be used for checking whether a given formula φ is a semantic property of a theory.

Remark 13 (Horn theories). *van Emden and Kowalski proved that Horn theories* Th *have the* model intersection property*: the intersection of Herbrand models of* Th *is also a model of* Th. *Thus,* $\mathcal{I}_{\mathsf{Th}}$ *is* the *least Herbrand model of* Th *and Proposition 10 can be seen as establishing* initiality *of* $\mathcal{I}_{\mathsf{Th}}$ *in* $Mod(\mathsf{Th})$.

Non-Horn sets of sentences Th may have a least Herbrand model as well, which (by Proposition 9) is $\mathcal{I}_{\mathsf{Th}}$. Furthermore, for theories Th with several Herbrand models of interest (as in Example 8), we are often able to make use of our approach by representing each model $\mathcal{H}_i$ as the initial model $\mathcal{I}_{\mathsf{Th}_i}$ of a Horn theory Th_i obtained from Th and $\mathcal{H}_i$. We discuss these issues in Section 4.4 below.

Corollary 14. *Every logical consequence* φ *of a Horn theory* Th *is a semantic property of* Th.

There are theories Th which are incomplete: some formulas φ cannot be proved and the negation $\neg\varphi$ cannot be proved either. In contrast, according to Definition 11 either φ or $\neg\varphi$ is a semantic property of Th. If $\mathcal{I}_{\mathsf{Th}} \models \neg\varphi$ holds, then $\mathcal{I}_{\mathsf{Th}} \models \varphi$ does not hold and we often say that φ is *not* a semantic property of Th.

Remark 15 (Negative literals). *Theories usually specify true instances of relations only (see [6]) and there is no way to derive negative information from them by logical reasoning. This means that using theorem proving to verify a semantic property* $\neg A$ *for a given atom* A *is usually impossible. In Section 5 we investigate alternative methods to achieve this goal.*

4. Application to logic-based systems and languages

In this section we consider some systems whose specification can be translated into a MS-FOL theory so that we can reason about them by using FOL.

4.1. Relational databases

In the relational model of databases [10], data are organized as collections of n-tuples on possibly different domains (see Example 2). Codd used FOL to model relational databases: tuples are represented by atoms of MS-FOL where only constant function symbols are used to denote components of the domains (as in Example 2). A data sublanguage to qualify the information stored in the data base and also to formulate queries is also described. As remarked by Nicolas and Gallaire [46], the set DB of atoms representing the tuples in a relational data base (which coincide with its ground consequences, i.e., $DB = DB^{\vdash}$) are more naturally viewed as specifying a (Herbrand) *interpretation* $\mathcal{I}_{DB} = \mathcal{H}_{\Omega}(DB^{\vdash}) = \mathcal{H}_{\Omega}(DB)$ and formulas φ representing queries, functional dependencies, (static) integrity constraints, data base properties, etc. [4, 10, 16], are treated as sentences to be evaluated (i.e., checked for satisfiability) with respect to the interpretation: $\mathcal{I}_{DB} \models \varphi$ [46, pages 42 & 44].

Example 16. *Sentence (5) could be thought of as an integrity constraint of the relational database in Example 2 guaranteeing that no student is left unattended.*

Remark 17 (Dynamic aspects). *Data bases have a dynamic ingredient: they change by means of updates and deletions of the stored information. An important aspect of the maintenance of a database is fulfilling* dependency conditions *and dynamic* integrity constraints*, i.e., conditions that the data should satisfy after database transactions. In this paper we pay no attention to these issues.*

4.2. Deductive databases

In deductive databases [48], stored knowledge is represented by means of atomic components (the *extensional* database, *EDB*), and also by a *deductive* component consisting of *derivation rules* (*intensional* database, *IDB*) [48, page 128]. The FO theory $DDB = EDB \cup IDB$ fully describes the database.

Example 18. *A new monadic predicate* `active` *with argument of sort Teacher is added to* Teach *to qualify teachers as active if they teach some student:*

```
active(X) <- teach(X,Y).
```

Again, Nicolas and Gallaire's approach uses a canonical Herbrand interpretation $\mathcal{I}_{DDB} = \mathcal{H}_{\Omega}(DDB^{\vdash})$ to solve formulas φ. Relational and deductive databases can be investigated as *logic programs*.

4.3. Logic programs

A (basic) logic program $\mathcal{P}$ consists of a set of Horn *clauses* usually written

$$A \leftarrow B_1, \ldots, B_n \tag{10}$$

for some $n \geq 0$, where $A, B_1, \ldots, B_n$ are *atoms*. We obtain a theory $\overline{\mathcal{P}}$ (or just $\mathcal{P}$ if no confusion arises) associated to $\mathcal{P}$ by just interpreting each clause $A \leftarrow B_1, \ldots, B_n$ as a sentence $(\forall \vec{x})\ B_1 \wedge \cdots \wedge B_n \Rightarrow A$.

$$
\begin{array}{rll}
(\forall y) & \mathsf{times}(0, y, 0) & (11)\\
(\forall u, x, y, z) & \mathsf{times}(x, y, u) \wedge \mathsf{add}(y, u, z) \Rightarrow \mathsf{times}(\mathsf{s}(x), y, z) & (12)\\
(\forall y) & \mathsf{add}(0, y, y) & (13)\\
(\forall x, y, z) & \mathsf{add}(x, y, z) \Rightarrow \mathsf{add}(\mathsf{s}(x), y, \mathsf{s}(z)) & (14)\\
(\forall x) & \mathsf{pow}(x, 0, \mathsf{s}(0)) & (15)\\
(\forall n, x, y, z) & \mathsf{pow}(x, n, y) \wedge \mathsf{times}(x, y, z) \Rightarrow \mathsf{pow}(x, \mathsf{s}(n), z) & (16)\\
 & \mathsf{even}(0) & (17)\\
(\forall x) & \mathsf{odd}(x) \Rightarrow \mathsf{even}(\mathsf{s}(x)) & (18)\\
 & \mathsf{odd}(\mathsf{s}(0)) & (19)\\
(\forall x) & \mathsf{even}(x) \Rightarrow \mathsf{odd}(\mathsf{s}(x)) & (20)
\end{array}
$$

Figure 1: FO theory for program in Example 19

Example 19. *Consider the following logic program:*

```
times(0,Y,0).
times(s(X),Y,Z) <- times(X,Y,U), add(Y,U,Z).
add(0,Y,Y).
add(s(X),Y,s(Z)) <- add(X,Y,Z).
pow(X,0,s(0)).
pow(X,s(N),Z) <- pow(X,N,Y), times(X,Y,Z).
even(0).
even(s(X)) <- odd(X).
odd(s(0)).
odd(s(X)) <- even(X).
```

Theory Powers *associated to this program is in Figure 1.*

Goals (or queries) G to be solved with respect to logic programs are sequences of atoms, written `?-`$A_1, \ldots, A_m$. What is proved is that the existential closure $(\exists \vec{x})\ A_1 \wedge \cdots \wedge A_m$ of the conjunction of the atoms holds in the least Herbrand model $\mathcal{I}_\mathcal{P}$, i.e., $\mathcal{I}_\mathcal{P} \models (\exists \vec{x})\ A_1 \wedge \cdots \wedge A_m$, possibly providing witnesses of it by means of appropriate substitutions of variables in the atoms.

Example 20. *A goal* `?-even(X)` *will be solved by a typical LP system by answering 'yes' and providing an instantiation of* `X` *which makes the atom true:* `X`/0*, then (if asked for more solutions)* `X`/s(s(0))*, etc.*

Although the most representative logic programming language (Prolog [11]) is untyped, other logic programming languages like Mercury [57] are typed and programs are naturally considered as many-sorted theories.

4.4. Answer Set Programming

Answer Set Programming (ASP, see [18] and the references therein), extends logic programming so that program clauses have a more general syntax,

including (classical and 'as-failure') negation, disjunction, arithmetic operations, choice, constraints, preferences, etc. For instance, in ASP Prolog, clauses are of the following form (compare with (10))

$$L_0 \; or \; \cdots \; or \; L_k \leftarrow L_{k+1}, \ldots, L_m, \; not \; L_{m+1}, \ldots, \; not \; L_n \tag{21}$$

where $L_0, \ldots, L_n$ are (many-sorted) literals and connectives *not* and *or* are called *negation as failure* or *default negation* and *epistemic disjunction* respectively [18]. Negation '$\neg$' occurring in literals is often called *classical negation*.

Remark 21 (ASP programs as FO theories). *Logic program clauses (10) can be viewed as first-order Horn clauses, and logic programs $\mathcal{P}$ as Horn theories $\overline{\mathcal{P}}$. In contrast, ASP clauses (21) are not immediately viewed as FO sentences: they include two different negation operators ('*not*' and $\neg$); also, epistemic disjunction '*or*' is not equivalent to FO connective $\vee$.*

ASP programs without classical negation or epistemic disjunction are often called *normal* logic programs (NLPs).[7] Program execution in ASP computes *answer sets*, which are "a possible set of beliefs of an agent associated to the program" see [18, page 288]. In contrast to (basic) logic programs, the collection of answer sets for ASP programs can be empty (there is no answer set) or contain (infinitely) many of them. Accordingly, "the answer to a query may depend on which answer set is selected".

Example 22. *When the non-Horn theory* $\mathtt{Th} = \{p(a) \vee p(b)\}$ *in Example 8 is viewed as an ASP program* $\mathcal{P} = \{$`p(a) or p(b)`$\}$*, the answer sets of $\mathcal{P}$ are $\mathcal{H} = \{p(a)\}$ and $\mathcal{H}' = \{p(b)\}$ in Example 8.*

When dealing with NLPs, answer sets are Herbrand interpretations.[8]

Example 23. *(cont. Example 22) The following NLP program $\mathcal{P}$ could be (equivalently) associated to* `Th` *in Example 8 [18, Section 7.3.4]:*

```
p(a) <- not p(b)
p(b) <- not p(a)
```

Its answer sets are $\mathcal{H}$ and $\mathcal{H}'$ in Example 8, see Example 60 below.

4.5. Rewriting-based systems

The insertion of a 'rewriting-based system' $\mathcal{R}$ (e.g., Term Rewriting Systems (TRSs, *Context-Sensitive* TRSs *Conditional* TRSs (CTRSs, *Membership Equational Programs* [32, 47, 41], and more general rewriting-based formalisms

[7] Classical negation can be removed from ASP programs by adding predicates p' associated to predicates p to obtain a *positive* form $p'(\vec{t})$ for literals $\neg p(\vec{t})$, see [18, Section 7.3.1].

[8] Answer sets of fully general ASP programs admit negative literals in (consistent) answer sets S. A *three valued logic* for knowledge representation is used: the truth value of atoms A in S is true; if $\neg A$ is in S, then A is false; otherwise the truth value of A is *unknown* [18].

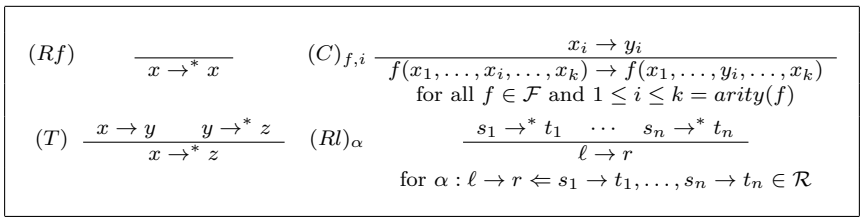

Figure 2: Inference rules for conditional rewriting with a CTRS $\mathcal{R}$ with signature $\mathcal{F}$

[3, 20, 42]) into FOL is made in two steps: first a specialized inference system $\mathfrak{I}(\mathcal{R})$ is obtained from the generic inference system $\mathfrak{I}$ describing the operational semantics of $\mathcal{R}$ as provability (à la *natural deduction*) of goals with predicate symbols $\to$ (one-step reduction), $\to^*$ (many-step reduction), $\downarrow$ (joinability), $\_ : s$ (membership for a given sort s), etc.; then, a set $\overline{\mathcal{R}}$ of sentences is obtained from $\mathfrak{I}(\mathcal{R})$ by just treating inference rules $\frac{B_1 \ \cdots \ B_n}{A}$ as universally quantified *implications* $(\forall \vec{x})\ B_1 \wedge \cdots \wedge B_n \Rightarrow A$ (with provability equivalently implemented now as resolution [54] or using Hilbert's style [40, Section 2.3]).

For instance, a (many-sorted) CTRS is a pair $\mathcal{R} = (\Sigma, R)$ where Σ is a many-sorted signature of function symbols and R is a set of conditional rules. A conditional rule $\ell \to r \Leftarrow c$ is a triple where ℓ and r are terms and the *conditional part* c is a sequence $s_1 \approx t_1, \ldots, s_n \approx t_n$ of expressions $s_i \approx t_i$. Such expressions are usually interpreted as *reachability* or *joinability* problems after an appropriate instantiation with a substitution σ, i.e., for all i, $1 \le i \le n$, $\sigma(s_i) \to^*_{\mathcal{R}} \sigma(t_i)$ (for the *rewriting semantics*); or $\sigma(s_i) \downarrow_{\mathcal{R}} \sigma(t_i)$ (for the joinability semantics[9]) [47]. We focus on the rewriting semantics. We write $s \to^*_{\mathcal{R}} t$ for terms s and t iff there is a proof tree for $s \to^* t$ using $\mathcal{R}$ in the inference system of Figure 2 (and similarly for one-step rewriting steps $s \to_{\mathcal{R}} t$ regarding *proofs* of the goal $s \to t$) [37].

Remark 24. Schematic *rules* $\frac{B_1 \ \cdots \ B_n}{A}$ *actually denote* instances $\frac{\sigma(B_1) \ \cdots \ \sigma(B_n)}{\sigma(A)}$ *by a substitution. For instance,* $(Rl)_\alpha$ *in Figure 2 establishes that, for all substitutions* σ*,* $\sigma(\ell)$ *rewrites into* $\sigma(r)$ *whenever* $\sigma(s_i) \to^* \sigma(t_i)$ *for* $1 \le i \le n$*.*

In the logic of CTRSs $\mathcal{R}$, with binary *predicates* $\to$ and $\to^*$, $\mathfrak{I}(\mathcal{R})$ is obtained from the inference rules in Figure 2 by *specializing* $(C)_{f,i}$ for each $f \in \mathcal{F}$ and $1 \le i \le ar(f)$, and $(Rl)_\alpha$ for all $\alpha : \ell \to r \Leftarrow c \in R$. Inference rules $\frac{B_1 \ \cdots \ B_n}{A}$ become universally quantified *implications* $B_1 \wedge \cdots \wedge B_n \Rightarrow A$ to obtain $\overline{\mathcal{R}}$.

Example 25. *Consider the following* Maude *[8] specification implementing the usual arithmetic operations over the naturals (sort* `N`*) together with function* `head`*, which returns the head of a list of natural numbers (sort* `LN`*):*

[9] A joinability condition $s \downarrow t$ is equivalent to a reachability condition $s \to^* x, t \to^* x$ if x is a fresh variable not occurring elsewhere in the rule.

$$
\begin{array}{rl}
(\forall x : \mathtt{N}) & x \to^* x \\
(\forall x : \mathtt{LN}) & x \to^* x \\
(\forall x, y, z : \mathtt{N}) & x \to y \wedge y \to^* z \Rightarrow x \to^* z \\
(\forall x, y, z : \mathtt{LN}) & x \to y \wedge y \to^* z \Rightarrow x \to^* z \\
(\forall x, y : \mathtt{N}) & x \to y \Rightarrow \mathsf{s}(x) \to \mathsf{s}(y) \\
(\forall x, y, z : \mathtt{N}) & x \to y \Rightarrow \mathsf{add}(x, z) \to \mathsf{add}(y, z) \\
(\forall x, y, z : \mathtt{N}) & x \to y \Rightarrow \mathsf{add}(z, x) \to \mathsf{add}(z, y) \\
(\forall x, y, z : \mathtt{N}) & x \to y \Rightarrow \mathsf{mul}(x, z) \to \mathsf{mul}(y, z) \\
(\forall x, y, z : \mathtt{N}) & x \to y \Rightarrow \mathsf{mul}(z, x) \to \mathsf{mul}(z, y) \\
(\forall x, y : \mathtt{N}, xs : \mathtt{LN}) & x \to y \Rightarrow \mathsf{cons}(x, xs) \to \mathsf{cons}(y, xs) \\
(\forall x : \mathtt{N}, xs, ys : \mathtt{LN}) & x \to y \Rightarrow \mathsf{cons}(x, xs) \to \mathsf{cons}(x, ys) \\
(\forall x, y : \mathtt{LN}) & x \to y \Rightarrow \mathsf{head}(x) \to \mathsf{head}(y) \\
(\forall x : \mathtt{N}) & \mathsf{add}(\mathsf{0}, x) \to x \\
(\forall x, y : \mathtt{N}) & \mathsf{add}(\mathsf{s}(x), y) \to \mathsf{s}(\mathsf{add}(x, y)) \\
(\forall x : \mathtt{N}) & \mathsf{mul}(\mathsf{0}, x) \to \mathsf{0} \\
(\forall x, y : \mathtt{N}) & \mathsf{mul}(\mathsf{s}(x), y) \to \mathsf{add}(y, \mathsf{mul}(x, y)) \\
(\forall x : \mathtt{N}, xs : \mathtt{LN}) & \mathsf{head}(\mathsf{cons}(x, xs)) \to x
\end{array}
$$

Figure 3: Horn theory for `ExAddMulHead` ($\to$ and $\to^*$ are overloaded)

```
mod ExAddMulHead is
  sorts N LN .
  op Z : -> N .        op suc : N -> N .  ops add mul : N N -> N .
  op head : LN -> N .  op nil : -> LN .   op cons : N LN -> LN .
  vars x y : N .       var xs : LN .
  rl add(Z,x) => x .   rl add(suc(x),y) => suc(add(x,y)) .
  rl mul(Z,x) => Z .   rl mul(suc(x),y) => add(y,mul(x,y)) .
  rl head(cons(x,xs)) => x .
endm
```

The MS-FO theory $\overline{\texttt{ExAddMulHead}}$ *for* `ExAddMulHead` *is in Figure 3.*

In the *canonical model* $\mathcal{I}_{\mathcal{R}} = \mathcal{H}(\overline{\mathcal{R}}^{\vdash})$ of $\overline{\mathcal{R}}$, $\to$ and $\to^*$ are interpreted as the sets $(\to)^{\mathcal{I}_{\mathcal{R}}}$ and $(\to^*)^{\mathcal{I}_{\mathcal{R}}}$ of pairs (s, t) of ground terms s and t such that $s \to_{\mathcal{R}} t$ and $s \to^*_{\mathcal{R}} t$, respectively. Semantic properties of CTRSs are referred to $\mathcal{I}_{\mathcal{R}}$.

5. Preservation of Many-Sorted First-Order Sentences

In the following, we consider sentences in *prenex* form[10]

$$(Q_1 x_1 : s_1) \cdots (Q_k x_k : s_k) \bigvee_{i=1}^{m} \bigwedge_{j=1}^{n_i} L_{ij} \tag{22}$$

where $M(x_1, \ldots, x_k) = \bigvee_{i=1}^{m} \bigwedge_{j=1}^{n_i} L_{ij}$ is often called the *matrix* of (22). Here, (a) for all $1 \leq i \leq m$ and $1 \leq j \leq n_i$, L_{ij} are *literals* $L_{ij} = A_{ij}$ or $L_{ij} = \neg A_{ij}$ for some atom A_{ij} (in the first case, we say that (the *sign* of) L_{ij} is *positive*; otherwise, it is *negative* [14]) (b) $x_1, \ldots, x_k$ for some $k \geq 0$ are the variables occurring in those literals (of sorts $s_1, \ldots, s_k$, respectively), and (c) $Q_1, \ldots, Q_k$ are universal/existential quantifiers. A sentence (22) is said to be *positive* if all literals are; if, additionally, $Q_q = \exists$ for all $1 \leq q \leq k$, then it is an ECBCA.

Remark 26 (The normal form (22)). *The* matrix $M(x_1, \ldots, x_k)$ *of* (22) *is presented as a disjunction of conjunctions (disjunctive normal form, DNF. By the distributive laws for* $\wedge$ *and* $\vee$*,* $M(x_1, \ldots, x_k)$ *can be written in* conjunctive normal form *(CNF)* $M'(x_1, \ldots, x_k) = \bigwedge_{i=1}^{m'} \bigvee_{j=1}^{n'_i} L'_{ij}$ *without changing the sign of the literals. Our results rely on the sequence of quantifiers and the sign of the literals, which remain unchanged if a CNF-based form*

$$(Q_1 x_1 : s_1) \cdots (Q_k x_k : s_k) \bigwedge_{i=1}^{m'} \bigvee_{j=1}^{n'_i} L'_{ij} \tag{23}$$

is used instead. Thus, using (22) or (23) is not essential.

The main result of this section is the following (where, following Section 2, h_{s_q} is the component of $h_{\mathcal{A}}$ for sort s_q).

Theorem 27. *Let* Ω *be a signature,* Th *be a set of ground atoms,* φ *be as (22), and* $\mathcal{A}$ *be a model of* Th *such that (a) for all* q*,* $1 \leq q \leq k$*, if* $Q_q = \forall$ *then* h_{s_q} *is surjective and (b) for all negative literals* $\neg P(\vec{t})$ *in* φ *and ground substitutions* σ*, if* $h(\sigma(\vec{t})) \in P^{\mathcal{A}}$ *then* $\sigma(\vec{t}) \in P^{\mathcal{I}_{\mathsf{Th}}}$*. Then,* $\mathcal{I}_{\mathsf{Th}} \models \varphi \Longrightarrow \mathcal{A} \models \varphi$*.*

Proof. Denote as $M(x_1, \ldots, x_k)$ the *matrix* formula $\bigvee_{i=1}^{m} \bigwedge_{j=1}^{n_i} L_{ij}$ (containing variables $x_1, \ldots, x_k$). We proceed by induction on k.

If $k = 0$, then each L_{ij} is a ground literal for all $1 \leq i \leq m$ and $1 \leq j \leq n_i$. If $\mathcal{I}_{\mathsf{Th}} \models M$, then there is i, $1 \leq i \leq m$ such that $\mathcal{I}_{\mathsf{Th}} \models L_{ij}$ for all $1 \leq j \leq n_i$. Let $\mathcal{A}$ be a model of Th. There is a unique homomorphism $h : \mathcal{I}_{\mathsf{Th}} \to \mathcal{A}$. Since L_{ij} is ground, whenever L_{ij} is positive, by definition of homomorphism, $\mathcal{A} \models L_{ij}$ holds. If $L_{ij} = \neg P(\vec{t})$ for some predicate $P : w$ and $\vec{t} \in \mathcal{T}_{\Sigma w}$, since $\mathcal{I}_{\mathsf{Th}} \models L_{ij}$, we have $\vec{t} \notin P^{\mathcal{I}}$; hence, since $\sigma(\vec{t}) = \vec{t}$, by condition (b), $h(\vec{t}) = [\vec{t}]^{\mathcal{A}} \notin P^{\mathcal{A}}$, i.e., $\mathcal{A} \models L_{ij}$. Thus, $\mathcal{A} \models L_{ij}$ for all $1 \leq j \leq n_i$ and $\mathcal{A} \models M$.

[10] Every sentence φ can be equivalently written in *prenex normal form.*

If $k > 0$, then if $\mathcal{I}_{\mathsf{Th}} \models (Q_1 x_1 : s_1)(Q_2 x_2 : s_2) \cdots (Q_k x_k : s_k) M(x_1, \ldots, x_k)$, we consider two cases according to Q_1.
(1) If $Q_1 = \exists$, then there is $t \in \mathcal{T}_{\Sigma s_1}$ such that

$$\mathcal{I}_{\mathsf{Th}} \models (Q_2 x_2 : s_2) \cdots (Q_k x_k : s_k) M(x_1, x_2, \ldots, x_k)[x_1 \mapsto t]$$

Hence $\mathcal{I}_{\mathsf{Th}} \models (Q_2 x_2 : s_2) \cdots (Q_k x_k : s_k) M(t, x_2, \ldots, x_k)$. By the induction hypothesis, $\mathcal{A} \models (Q_2 x_2 : s_2) \cdots (Q_k x_k : s_k) M(t, x_2, \ldots, x_k)$. Thus, there is an element $b = t^{\mathcal{A}} \in \mathcal{A}$ such that

$$\mathcal{A} \models (Q_2 x_2 : s_2) \cdots (Q_k x_k : s_k) M(x_1, x_2, \ldots, x_k)[x_1 \mapsto b]$$

and hence $\mathcal{A} \models (Q_1 x_1 : s_1) \cdots (Q_k x_k : s_k)) M(x_1, \ldots, x_k)$ as well.
(2) If $Q_1 = \forall$, then for all $t \in \mathcal{T}_{\Sigma s_1}$, we have $\mathcal{I}_{\mathsf{Th}} \models (Q_2 x_2 : s_2) \cdots (Q_k x_k : s_k) M(t, x_2, \ldots, x_k)$. By the induction hypothesis, for each such terms t, $\mathcal{A} \models (Q_2 x_2 : s_2) \cdots (Q_k x_k : s_k) M(t, x_2, \ldots, x_k)$. Since $h : \mathcal{T}_{\Sigma s_1} \to \mathcal{A}_{s_1}$ is surjective, for all $b \in \mathcal{A}_{s_1}$ there is $t \in \mathcal{T}_{\Sigma}$ such that $h(t) = b$. Thus,

$$\mathcal{A} \models (Q_2 x_2 : s_2) \cdots (Q_k x_k : s_k) M(x_1, x_2, \ldots, x_k)[x_1 \mapsto b]$$

for all $b \in \mathcal{A}_{s_1}$, i.e., $\mathcal{A} \models (Q_1 x_1 : s_1) \cdots (Q_k x_k : s_k) M(x_1, x_2, \ldots, x_k)$. □

In the following, unless stated otherwise, Th is a theory and φ is a sentence (22), whose notation is often referred in the statements of our results.

Corollary 28. *Let $\mathcal{A}$ be a model of Th such that (a) for all q, $1 \leq q \leq k$, if $Q_q = \forall$ then h_{s_q} is surjective and (b) for all negative literals $\neg P(\vec{t})$ in φ and ground substitutions σ, if $h(\sigma(\vec{t})) \in P^{\mathcal{A}}$ then $\sigma(\vec{t}) \in P^{\mathcal{I}_{\mathsf{Th}}}$. If $\mathcal{A} \models \neg\varphi$, then $\mathcal{I}_{\mathsf{Th}} \models \neg\varphi$.*

Proof. Proposition 10 and Theorem 27. □

Models $\mathcal{A}$ in Corollary 28 can be generated from Th and φ by using tools like AGES [23] or Mace4. Note that, for ECBCA formulas, conditions (a) and (b) vacuously hold. Thus, as already established in [25, Theorem 2.4.3(a)], homomorphisms always preserve ECBCA formulas.

Example 29. *Consider the theory* Powers *for the logic program in Example 19. The existence of a (natural) number which is the n^{th} root of 3 (for some $n > 1$) is claimed by the following (ECBCA) sentence:*

$$(\exists x)(\exists n)\ \mathsf{pow}(x, \mathsf{s}(\mathsf{s}(n)), \mathsf{s}(\mathsf{s}(\mathsf{s}(\mathsf{0})))) \tag{24}$$

With Mace4 *we obtain a model $\mathcal{A}$ of* Powers $\cup \{\neg(24)\}$. *By Corollary 28, we conclude that (24) is not a semantic property of* Powers.

Remark 30 (Verification as satisfiability). *We can verify that φ is an Th-property by satisfiability as follows: (i) let $\overline{\varphi}$ be $\neg\varphi$; then (ii) find a structure $\mathcal{A}$ satisfying (a) and (b) with regard to $\overline{\varphi}$ and such that (iii) $\mathcal{A} \models \mathsf{Th} \cup \{\varphi\}$ (note that $\neg\overline{\varphi}$ and φ coincide). By Corollary 28, $\mathcal{I}_{\mathsf{Th}} \models \varphi$ holds.*

We postpone the treatment of condition (a) in the previous results (surjectivity of h_s for a given sort s) to Section 6. In the following section, we discuss condition (b) about dealing with negative literals in (22).

5.1. Dealing with negative literals

Given a predicate $P \in \Pi_w$, let $N_{\mathsf{Th}}(P) = \mathcal{I}_w - P^{\mathcal{I}_{\mathsf{Th}}}$ be the tuples in $\mathcal{T}_{\Sigma w}$ obtained as the complement of the interpretation of P in $\mathcal{I}_{\mathsf{Th}}$. Let $\mathcal{N}_{\mathsf{Th}}(P) = \{\neg P(\vec{t}) \mid \vec{t} \in N_{\mathsf{Th}}(P)\}$ be the corresponding tuples viewed as negative literals.

Remark 31. *In general, $\mathcal{N}_{\mathsf{Th}}(P)$ is infinite and incomputable. We can provide a* finite description *of $\mathcal{N}_{\mathsf{Th}}(P)$ for $P \in \Pi_w$, though, if w is a sequence $s_1 \cdots s_n$ of sorts such that $\mathcal{T}_{\Sigma s_i}$ is finite for all $1 \leq i \leq n$ (for instance, dealing with relational databases, see Example 34).*

In the following, we let $\mathsf{N}_{\mathsf{Th},\varphi} = \bigcup_{\neg P(\vec{t}) \text{ in } \varphi} \mathcal{N}_{\mathsf{Th}}(P)$.

Proposition 32. *Let $\mathcal{A}$ be a model of $\mathsf{Th} \cup \mathsf{N}_{\mathsf{Th},\varphi}$, $\neg P(\vec{t})$ be a negative literal in φ and σ be a ground substitution. If $h_{\mathcal{A}}(\sigma(\vec{t})) \in P^{\mathcal{A}}$, then $\sigma(\vec{t}) \in P^{\mathcal{I}_{\mathsf{Th}}}$.*

Proof. If $\sigma(\vec{t}) \notin P^{\mathcal{I}_{\mathsf{Th}}}$ then $\neg P(\sigma(\vec{t})) \in \mathcal{N}_{\mathsf{Th}}(P) \subseteq \mathsf{N}_{\mathsf{Th},\varphi}$. Since $\mathcal{A} \models \mathsf{Th} \cup \{\neg P(\sigma(\vec{t}))\}$, by Proposition 10 $h(\sigma(\vec{t})) \notin P^{\mathcal{A}}$. □

By using Proposition 32, we recast Corollary 28 as follows:

Corollary 33. *Let $\mathcal{A}$ be a model of $\mathsf{Th} \cup \mathsf{N}_{\mathsf{Th},\varphi}$ such that for all q, $1 \leq q \leq k$, if $Q_q = \forall$ then h_{s_q} is surjective. If $\mathcal{A} \models \neg\varphi$, then $\mathcal{I}_{\mathsf{Th}} \models \neg\varphi$.*

Example 34. *According to Remark 30, we can prove that (5) is a semantic property of* Teach *by first obtaining its negation $\neg(5)$, i.e.,*

$$(\exists y : \mathit{Student})(\forall x : \mathit{Teacher})\ \neg\mathtt{teach}(x, y) \tag{25}$$

and then applying Corollary 33 to (25). Since

$$N(\mathtt{teach}) = \{(t, s) \mid t \in \{\mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{d}\}, s \in \{\mathtt{p}, \mathtt{q}, \mathtt{r}\}\} - \{(\mathtt{a}, \mathtt{p}), (\mathtt{a}, \mathtt{q}), (\mathtt{b}, \mathtt{q}), (\mathtt{c}, \mathtt{r})\}$$

we have to consider (the proof finishes in Example 38 below):

$$\begin{aligned}\mathsf{N}_{\mathsf{Teach},(25)} = \{&\neg\mathtt{teach}(\mathtt{a}, \mathtt{r}), \neg\mathtt{teach}(\mathtt{b}, \mathtt{p}), \neg\mathtt{teach}(\mathtt{b}, \mathtt{r}), \neg\mathtt{teach}(\mathtt{c}, \mathtt{p}),\\ &\neg\mathtt{teach}(\mathtt{c}, \mathtt{q}), \neg\mathtt{teach}(\mathtt{d}, \mathtt{p}), \neg\mathtt{teach}(\mathtt{d}, \mathtt{q}), \neg\mathtt{teach}(\mathtt{d}, \mathtt{r})\}\end{aligned}$$

5.2. Removing sort information if necessary

In some cases, removing sort information from the original signature $\Omega = (S, \Sigma, \Pi)$ can be useful (or necessary) to use model generators which do not support sorts (e.g., Mace4). Let $\Omega^{\downarrow} = (\{\mathrm{s}\}, \Sigma^{\downarrow}, \Pi^{\downarrow})$ be a new, one-sorted signature where s is a dummy sort. Each $f \in \Sigma_{w,s}$ for $w = s_1 \cdots s_n \in S^*$ becomes an n-ary symbol $f^{\downarrow} : \mathrm{s} \cdots \mathrm{s} \to \mathrm{s}$, and each $P \in \Pi_w$ becomes an n-ary predicate $P^{\downarrow}$ (overloaded symbols collapse into a single symbol). For each $s \in S$, terms $t \in \mathcal{T}_{\Sigma s}$ inductively become terms $t^{\downarrow} \in \mathcal{T}_{\Sigma^{\downarrow}}$ and similarly for atoms and more complex formulas φ. In this way, given a set of atoms Th, we obtain an interpretation $\mathcal{I}^{\downarrow}_{\mathsf{Th}} = \mathcal{H}_{\Omega^{\downarrow}}(\mathsf{Th}^{\downarrow})$ from $\mathcal{I}_{\mathsf{Th}}$. Note that $\mathcal{I}^{\downarrow}_{\mathsf{Th}}$ is also an Ω-structure.

In the following result, which is a consequence of Theorem 27, $h : \mathcal{I}_{\mathsf{Th}} \to \mathcal{I}^{\downarrow}_{\mathsf{Th}}$ is the (S-sorted) interpretation Ω-homomorphism $h(t) = t^{\downarrow}$, and $h^{\downarrow}_{\mathcal{A}} : \mathcal{I}^{\downarrow}_{\mathsf{Th}} \to \mathcal{A}$ (or just $h^{\downarrow}$) is the interpretation $\Omega^{\downarrow}$-homomorphism mapping 'desorted' terms in $\mathcal{T}_{\Sigma^{\downarrow}}$ into elements in the domain of the $\Omega^{\downarrow}$-structure $\mathcal{A}$.

Corollary 35. *Let Ω be a signature, Th be an Ω-theory, φ be as (22), and $\mathcal{A}$ be an $\Omega^{\downarrow}$-structure such that (a) for all q, $1 \leq q \leq k$, if $Q_q = \forall$, then $h^{\downarrow} \circ h_{s_q} : \mathcal{I}_{s_q} \to \mathcal{A}$ is surjective and (b) for all $L_{ij} = \neg P(\vec{t})$ and ground substitutions σ, if $h^{\downarrow}(h(\sigma(\vec{t}))) \in (P^{\downarrow})^{\mathcal{A}}$, then $\sigma(\vec{t}) \in P^{\mathcal{I}}$. If $\mathcal{A} \models \mathsf{Th}^{\downarrow} \cup \{\neg\varphi^{\downarrow}\}$, then $\mathcal{I}_{\mathsf{Th}} \models \neg\varphi$.*

Example 36. *(cont. Example 5) Semantic property (4) of* Teach *is verified with* Mace4 *by computing a model of* $\mathsf{Teach}^{\downarrow} \cup \{(4)^{\downarrow}\}$*, where* $(4)^{\downarrow}$ *is*

$$\neg(\exists y : \mathrm{S})\, \mathtt{teach}(\mathtt{d}, y) \quad \textit{or just} \quad \neg(\exists y)\, \mathtt{teach}(\mathtt{d}, y) \tag{26}$$

Proposition 37. *Let Ω be a signature, Th be a set of ground atoms, φ be as (22), and $\mathcal{A}$ be a model of $\mathsf{Th}^{\downarrow} \cup \mathsf{N}^{\downarrow}_{\mathsf{Th},\varphi}$. Let $\neg P(\vec{t})$ be a negative literal in φ and σ be a ground substitution. If $h^{\downarrow}(h(\sigma(\vec{t}))) \in (P^{\downarrow})^{\mathcal{A}}$, then $\sigma(\vec{t}) \in P^{\mathcal{I}}$.*

In general, $h_s : \mathcal{T}_{\Sigma s} \to \mathcal{T}_{\Sigma^{\downarrow}}$ is *not* surjective: after collapsing all sorts into a single one, many terms in $\mathcal{T}_{\Sigma^{\downarrow}}$ have no corresponding well-formed term in $\mathcal{T}_{\Sigma s}$. Thus, in order to guarantee surjectivity of $h^{\downarrow} \circ h_s$, requiring surjectivity of $h^{\downarrow}$ does not suffice. We need to impose more stringent conditions (see Section 6).

Example 38. *(cont. Example 34) We failed to obtain a model of* $\mathsf{Teach} \cup \mathsf{N}_{\mathsf{Teach},(25)} \cup \{(5)\}$ *with* AGES*. However,* Mace4 *can be used to verify (5) by using Corollary 35 for the unsorted version*

$$(\forall y)(\exists x)\, \mathtt{teach}(x, y) \tag{27}$$

of (5). The negation of (27) is

$$(\exists y)(\forall x)\, \neg\mathtt{teach}(x, y) \tag{28}$$

With Mace4 *we obtain a model $\mathcal{A}$ of $\mathsf{Teach}^{\downarrow} \cup \mathsf{N}^{\downarrow}_{\mathsf{Teach},(25)} \cup \{(27)\}$ with $\mathcal{A} = \{0, 1, 2, 3\}$; $\mathtt{a}^{\mathcal{A}} = \mathtt{p}^{\mathcal{A}} = 0$, $\mathtt{b}^{\mathcal{A}} = \mathtt{q}^{\mathcal{A}} = 1$, $\mathtt{c}^{\mathcal{A}} = \mathtt{r}^{\mathcal{A}} = 2$, and $\mathtt{d}^{\mathcal{A}} = 3$; finally $\mathtt{teach}^{\mathcal{A}} = \{(0,0), (0,1), (1,1), (2,2)\}$. Since only constant function symbols are involved, $h^{\downarrow} \circ h_{Teacher}$ is surjective.*

Note that, when using Corollary 35, sort information in φ is somehow preserved due to the *local* requirements of surjectivity for $h^{\downarrow} \circ h_s$ (for sort s only) and also considering negative atoms $\neg P(\vec{t})^{\downarrow} \in \mathsf{N}^{\downarrow}_{\mathsf{Th},\varphi}$ (usually strictly included in $\mathsf{N}_{\mathsf{Th}^{\downarrow},\varphi^{\downarrow}}$).

Example 39. *(cont. Example 38) Note that (27) is* not *a semantic property of $\mathsf{Teach}^{\downarrow}$: after clearing sorts from the signature, all constants represent the same kind of objects. In particular, no constant k satisfies $\mathtt{teach}^{\downarrow}(k, \mathtt{a})$.*

Many-sorted sentences can be recasted as unsorted sentences by treating sorts $s \in S$ as new monadic predicates π_s [63] leading to an alternative (more complex) sort elimination procedure. For instance, (5) would be written as follows:

$$(\forall y)(\exists x)\ \mathit{Student}(y) \Rightarrow \mathit{Teacher}(x) \wedge \mathtt{teach}(x, y) \tag{29}$$

(instead of (27)) and Teach would be extended to an equivalent unsorted theory UTeach by adding appropriate atoms to explicitly give sorts to constant symbols:

```
teacher(a).   teacher(b).   teacher(c).   teacher(d).
student(p).   student(q).   student(r).
```

Note, however, that the normalized version of (29)

$$(\forall y)(\exists x)\, \neg \mathit{Student}(y) \vee (\mathit{Teacher}(x) \wedge \mathtt{teach}(x, y)) \tag{30}$$

is structurally different from (5): it contains a negative literal. Thus, we often better remove sorts as to use Corollary 35 instead of Corollary 28.

6. Surjective homomorphisms

Given a signature $\Omega = (S, \Sigma, \Pi)$, $s \in S$, and an Ω-structure $\mathcal{A}$, ensuring *surjectivity* of $h : \mathcal{T}_{\Sigma s} \to \mathcal{A}_s$ is important to use our results. We investigate how to guarantee *surjectivity* of h by satisfiability of an appropriate *theory* H. Although condition $(\forall x : s) \bigvee_{t \in \mathcal{T}_{\Sigma s}} x = t$ would do the work, if $\mathcal{T}_{\Sigma s}$ is infinite the obtained infinite disjunction is disallowed in FOL. We investigate appropriate refinements: for $\mathcal{A}_s$ finite, and for the general case (Section 6.2).

6.1. Structures with finite domains

Given a set $T \subseteq \mathcal{T}_{\Sigma s}$ of ground terms, consider the following sentences:

$$(\forall x : s) \bigvee_{t \in T} x = t \tag{31}$$

$$\bigwedge_{t,u \in T, t \neq u} \neg(t = u) \tag{32}$$

In the following, we write $(31)_s$ to make sort s referred in (31) explicit. We do the same with (32) and similar formulas below.

Proposition 40. *Let $\mathcal{A}$ be a model of Th, $s \in S$, and $T \subseteq \mathcal{T}_{\Sigma s}$. (a) If $\mathcal{A} \models (31)_s$, then h_s is surjective and $|\mathcal{A}_s| \leq |T|$. (b) If $\mathcal{A} \models (32)$, then $|\mathcal{A}_s| \geq |T|$.*

Proof. By Proposition 10, $h : \mathcal{I}_{\mathsf{Th}} \to \mathcal{A}$ is the unique homomorphism. For (a), we consider two cases. If $T = \emptyset$, then $\mathcal{A} \models (31)_s$ holds only if $\mathcal{A}_s = \emptyset$. Thus, h is trivially surjective and $|\mathcal{A}_s| = |T|$. If $T \neq \emptyset$, then surjectivity follows because for all $x \in \mathcal{A}_s$ we have $x = t^{\mathcal{A}} = h(t)$ for some $t \in T$ due to (31), i.e., cardinality of $\mathcal{A}_s$ cannot exceed that of T. For (b), since (32) imposes that distinct terms have different interpretations, we need at least $|T|$ elements in $\mathcal{A}_s$ to interpret terms in T, i.e., $|\mathcal{A}_s| \geq |T|$. □

Due to Proposition 40(a), denote $(31)_s$ as $\mathsf{SuH}^T_s(\Omega)$ (or just SuH^T_s or SuH^T if no confusion arises).

Example 41. *(cont. Example 2) For $T = \{\mathsf{p}, \mathsf{q}, \mathsf{r}\}$, $\mathsf{SuH}^T_{Student}$ is*

$$(\forall x : \mathit{Student})\ x = \mathsf{p} \vee x = \mathsf{q} \vee x = \mathsf{r} \tag{33}$$

In contrast to the situation discussed in the second part of Example 2, no model of $\mathsf{Teach} \cup \mathsf{SuH}^T_{Student} \cup \{\neg(5)\}$ is obtained now.

Example 42. *(cont. Example 38) For* $T' = \{\mathsf{a}, \mathsf{b}, \mathsf{c}, \mathsf{d}\}$*, we obtain the model of*

$$\mathsf{Teach} \cup \mathsf{SuH}^{T'}_{Teacher} \cup \mathsf{N}_{(25)} \cup \{(27)\} \tag{34}$$

displayed in Example 38. The use of $\mathsf{SuH}^{T'}_{Teacher}$ *guarantees surjectivity of* $h^{\downarrow} \circ h_{Teacher}$ *as required by Corollary 35.*

The following example illustrates the need of an alternative approach when sorts with an infinite number of ground terms are used.

Example 43. *With regard to program* `ExAddMulHead` *in Example 25, we express the claim of* `add` *being* commutative *as follows:*

$$(\forall x : \mathtt{N})(\forall y : \mathtt{N})(\exists z : \mathtt{N})\ \mathtt{add}(x, y) \to^* z \wedge \mathtt{add}(y, x) \to^* z \tag{35}$$

with $\to^*$ *the* many-step *rewriting predicate. For* $\overline{\mathtt{ExAddMulHead}}$ *in Figure 3 and* $T = \{0, \mathsf{head}(\mathsf{nil})\}$*, with* AGES *we obtain a model* $\mathcal{A}$ *of*

$$\overline{\mathtt{ExAddMulHead}} \cup \mathsf{SuH}^{T}_{\mathtt{N}} \cup \{\neg(35)\} \tag{36}$$

Sorts are interpreted by $\mathcal{A}_{\mathtt{N}} = \mathcal{A}_{\mathtt{LN}} = \{0, 1\}$*. For function and predicate symbols:*

$$\begin{array}{cccc} 0^{\mathcal{A}} = 1 & \mathsf{nil}^{\mathcal{A}} = 0 & \mathsf{suc}^{\mathcal{A}}(x) = x & \mathsf{add}^{\mathcal{A}}(x, y) = y \\ \mathsf{mul}^{\mathcal{A}}(x, y) = 1 & \mathsf{cons}^{\mathcal{A}}(x, xs) = x & \mathsf{head}^{\mathcal{A}}(xs) = xs & \\ x \to^{\mathcal{A}}_{\mathtt{N}} y \Leftrightarrow x = y & x(\to^*_{\mathtt{N}})^{\mathcal{A}} y \Leftrightarrow x = y & x \to^{\mathcal{A}}_{\mathtt{LN}} y \Leftrightarrow x = y & x(\to^*_{\mathtt{LN}})^{\mathcal{A}} y \Leftrightarrow \mathit{true} \end{array}$$

This proves that property (35) does not *hold in* `ExAddMulHead`*.*

Remark 44 (The choice of terms in T). *We failed to obtain a model of (36) with sets* T without $\mathsf{head}(\mathsf{nil})$*. In particular, with* $T_n = \{0, \mathsf{suc}(0), \ldots, \mathsf{suc}^n(0)\}$ *we obtained no model for any attempted* $n \geq 0$*. Thus, the* choice *of* T *can be crucial when using* SuH^{T}_{s}*. Not only the* number *but also the specific* shape *of terms in* T *can be important (possibly depending on the focused property).*

In the following, we investigate correct approaches to *avoid* such a *choice*.

6.2. Structures with arbitrary (possibly infinite) domains

Consider the following sentence:

$$(\forall x : s)(\exists n : Nat)\ term_s(x, n) \tag{37}$$

where *Nat* is a new sort, to be interpreted as the set $\mathbb{N}$ of natural numbers, and $term_s : s\ Nat$ is a new predicate for each $s \in S$. The intended meaning of $(37)_s$ is that, for all $x \in \mathcal{A}_s$, there is $t \in \mathcal{T}_{\Sigma s}$ of height at most n such that $x = t^{\mathcal{A}}$. We substantiate this, for each sort $s \in S$, by means of two (families of) formulas:

$$(\forall x : s)\ term_s(x, 0) \quad \Rightarrow \quad \bigvee_{c \in \Sigma_{\lambda, s}} x = c \tag{38}$$

$$(\forall x : s)(\forall n : Nat)(\exists m : Nat)\ (n > 0 \wedge term_s(x,n)) \Rightarrow$$

$$n > m \wedge \left(term_s(x,m) \vee \bigvee_{\substack{f \in \Sigma_{w,s} \\ w \in S^+}} (\exists \vec{y} : w) \left(x = f(\vec{y}) \wedge \bigwedge_{s_i \in w} terms_{s_i}(y_i, m) \right) \right) \tag{39}$$

By $(38)_s$, values x satisfying $term_s(x,0)$ will be represented by a constant c of sort s. By $(39)_s$, values x satisfying $term_s(x,n)$ for some $n > 0$ will be represented by some ground term u of height $m < n$, or by a term $t = f(t_1, \ldots, t_k)$, where f has rank $w \to s$ for some $w \in S^+$ and $t_1, \ldots, t_k$ have height m at most.

The set $K(s)$ of *s-relevant* sorts is the least set satisfying: (i) $s \in K(s)$ and (ii) if $f \in \Sigma_{s_1 \cdots s_k, s'}$ and $s' \in K(s)$, then $\{s_1, \ldots, s_n\} \subseteq K(s)$.

Example 45. *For* `ExAddMulHead` *in Example 25,* $K(\mathtt{N}) = \{\mathtt{N}, \mathtt{LN}\}$ *due to*

```
op head : LN -> N .
```

whose argument is of sort `LN`.

Let $\Omega_{Nat,s} = (S_{Nat}, \Sigma_{Nat}, \Pi_{Nat,K(s)})$ be an *extension* of Ω where $S_{Nat} = S \cup \{Nat\}$, Σ_{Nat} extends Σ with a new constant $\mathsf{0} : \lambda \to Nat$, and $\Pi_{Nat,K(s)}$ extends Π with $> : Nat\ Nat$ and a predicate $term_{s'} : s'\ Nat$ for each $s' \in K(s)$. We let

$$\mathsf{SuH}_s \quad = \quad \{(37)_{s'}, (38)_{s'}, (39)_{s'} \mid s' \in K(s)\} \tag{40}$$

Proposition 46. *Let $s \in S$ and $\mathcal{A}$ be an $\Omega_{Nat,s}$-structure which is a model of* Th. *Assume that $\mathcal{A}_{Nat} = \mathbb{N}$, $\mathsf{0}^{\mathcal{A}} = 0$, and $\mathsf{m} >^{\mathcal{A}} \mathsf{n} \Leftrightarrow \mathsf{m} >_{\mathbb{N}} \mathsf{n}$ for all $\mathsf{m}, \mathsf{n} \in \mathcal{A}_{Nat}$. If $\mathcal{A} \models \mathsf{SuH}_s$, then $h_{s'} : \mathcal{T}_{\Sigma s'} \to \mathcal{A}_{s'}$ is surjective for all $s' \in K(s)$.*

Proof. We have to prove that for all $\mathsf{x} \in \mathcal{A}_{s'}$, there is $t \in \mathcal{T}_{\Sigma s'}$ such that $\mathsf{x} = h(t)$. Since $\mathcal{A} \models (37)_{s'}$ holds, and due to the restrictions imposed to the interpretation of sort *Nat*, there is $\mathsf{n} \in \mathcal{A}^{Nat}$ such that

$$\mathcal{A} \models term_{s'}(x,n)[x \mapsto \mathsf{x}, n \mapsto \mathsf{n}] \tag{41}$$

We prove by complete induction on n that there is $t \in \mathcal{T}_{\Sigma s'}$ such that $\mathsf{x} = h(t)$. For the base case, if $\mathsf{n} = \mathsf{n}_0$ is the least element of $\mathcal{A}_{Nat}$, then, since $\mathcal{A} \models (38)_{s'}$ holds, and (41) holds, there is a constant symbol c of sort s' such that $\mathsf{x} = c^{\mathcal{A}}$, as required. If $\mathsf{n} > \mathsf{n}_0$, then $\mathcal{A} \models (39)'_{s'}$ and together with (41) there is m such that $\mathsf{n} > \mathsf{m}$ such that we have two possibilities: If $\mathcal{A} \models term_{s'}(x,n)[y \mapsto \mathsf{x}, n \mapsto \mathsf{m}]$, then, by the induction hypothesis the conclusion follows.

If $\mathcal{A} \models \left(\bigvee_{\substack{f \in \Sigma_{w,s'} \\ w \in S^+}} (\exists \vec{y})\, \big(x = f(\vec{y}) \wedge \bigwedge_{s_i \in w} terms_{s_i}(y_i, m)\big) \right)[x \mapsto \mathsf{x}, m \mapsto \mathsf{m}]$, then there is a function symbol $f : s_1 \cdots s_k \to s'$ and values $\mathsf{y}_i \in \mathcal{A}_{s_i}$, $1 \leq i \leq k$ such that $\mathsf{x} =_{\mathcal{A}_{s'}} f^{\mathcal{A}}(\mathsf{y}_1, \ldots, \mathsf{y}_k)$ and $\mathcal{A} \models terms_{s_i}(y_i, m)[y_i \mapsto \mathsf{y}_i, m \mapsto \mathsf{m}]$ holds for all $1 \leq i \leq k$. By the induction hypothesis, there are terms $t_i \in \mathcal{T}_{\Sigma s_i}$ such that $t_i^{\mathcal{A}} = \mathsf{y}_i$ for all $1 \leq i \leq k$. Therefore, for $t = f(t_1, \ldots, t_k)$, we have $h(f(t_1, \ldots, t_k)) = [f(t_1, \ldots, t_k)]^{\mathcal{A}} = f^{\mathcal{A}}(\mathsf{y}_1, \ldots, \mathsf{y}_k) = \mathsf{x}$ as required. □

Given an extension Ω' of Ω, every Ω'-structure $\mathcal{A}'$ defines an Ω-structure $\mathcal{A}$: just take $\mathcal{A}_s = \mathcal{A}'_s$ for all $s \in S$, and then $f^{\mathcal{A}}_{w,s} = f^{\mathcal{A}'}_{w,s}$ and $P^{\mathcal{A}}_w = P^{\mathcal{A}'}_w$ for all $w \in S^*$, $s \in S$, $f \in \Sigma_{w,s}$, and $P \in \Pi_w$. Thus, Proposition 46 is used to guarantee surjectivity of $h : \mathcal{T}_{\Sigma s'} \to \mathcal{A}_{s'}$, rather than $h : \mathcal{T}_{\Sigma_{Nat} s'} \to \mathcal{A}_{s'}$. By Propositions 40 and 46, and assuming H_φ to be an appropriate (maybe empty) version of SuH to be used with φ, we finally recast Corollary 33 as follows:

Corollary 47 (Satisfiability criterion). *Let $\mathcal{A}$ be a structure.*

$$\textit{If } \mathcal{A} \models \mathsf{Th} \cup \mathsf{H}_\varphi \cup \mathsf{N}_\varphi \cup \{\neg\varphi\}, \textit{ then } \mathcal{I}_{\mathsf{Th}} \models \neg\varphi.$$

Or, taking into account Remark 30,

$$\textit{If } \mathcal{A} \models \mathsf{Th} \cup \mathsf{H}_{\neg\varphi} \cup \mathsf{N}_{\neg\varphi} \cup \{\varphi\}, \textit{ then } \mathcal{I}_{\mathsf{Th}} \models \varphi.$$

Example 48. *We disprove (35) by using Corollary 47 by obtaining a model of*

$$\overline{\mathtt{ExAddMulHead}} \cup \mathsf{SuH}_{\mathtt{N}} \cup \{\neg(35)\} \tag{42}$$

Since $K(\mathtt{N}) = \{\mathtt{N}, \mathtt{LN}\}$ (Example 45), $\mathsf{SuH}_{\mathtt{N}}$ consists of the following sentences:

$$\begin{array}{l}
(\forall x : \mathtt{N})(\exists n : Nat)\ term_{\mathtt{N}}(x, n) \\
(\forall x : \mathtt{N})\ term_{\mathtt{N}}(x, 0) \Rightarrow x = \mathsf{0} \\
(\forall x : \mathtt{N})(\forall n : Nat)(\exists m : Nat)(\exists y : \mathtt{N})(\exists z : \mathtt{N})(\exists ys : \mathtt{LN}) \\
\quad n > 0 \wedge term_{\mathtt{N}}(x, n) \Rightarrow n > m \wedge [term_{\mathtt{N}}(x, m) \vee \\
\quad\quad (term_{\mathtt{N}}(y, m) \wedge term_{\mathtt{N}}(z, m) \wedge term_{\mathtt{LN}}(ys, m) \wedge \\
\quad\quad\quad (x = \mathsf{suc}(y) \vee x = \mathsf{add}(y, z) \vee x = \mathsf{mul}(y, z) \vee x = \mathsf{head}(ys)))]
\end{array}$$

for sort $\mathtt{N}$, together with the following ones for sort $\mathtt{LN}$:

$$\begin{array}{l}
(\forall xs : \mathtt{LN})(\exists n : Nat)\ term_{\mathtt{LN}}(xs, n) \\
(\forall xs : \mathtt{LN})\ term_{\mathtt{LN}}(xs, 0) \Rightarrow xs = \mathsf{nil} \\
(\forall xs : \mathtt{LN})(\forall n : Nat)(\exists m : Nat)(\exists y : \mathtt{N})(\exists ys : \mathtt{LN}) \\
\quad n > 0 \wedge term_{\mathtt{N}}(x, n) \Rightarrow n > m \wedge [term_{\mathtt{N}}(x, m) \vee \\
\quad\quad (term_{\mathtt{N}}(y, m) \wedge term_{\mathtt{LN}}(ys, m) \wedge xs = \mathsf{cons}(y, ys))]
\end{array}$$

With AGES we obtain a model $\mathcal{A}$ of (42). Sorts are interpreted as $\mathcal{A}_{\mathtt{N}} = \{0, 1\}$, $\mathcal{A}_{\mathtt{LN}} = \{-1, 0\}$, and $\mathcal{A}_{Nat} = \mathbb{N}$. For function and predicate symbols, we have:

$$\begin{array}{cccc}
\mathsf{Z}^{\mathcal{A}} = 0 & \mathsf{nil}^{\mathcal{A}} = -1 & \mathsf{suc}^{\mathcal{A}}(x) = x & \mathsf{add}^{\mathcal{A}}(x, y) = y \\
\mathtt{mul}^{\mathcal{A}}(x, y) = 0 & \mathtt{cons}^{\mathcal{A}}(x, xs) = -x & \mathtt{head}^{\mathcal{A}}(xs) = -xs & \\
x \to^{\mathcal{A}}_{\mathtt{N}} y \Leftrightarrow x = y & x(\to^*_{\mathtt{N}})^{\mathcal{A}} y \Leftrightarrow x = y & x \to^{\mathcal{A}}_{\mathtt{LN}} y \Leftrightarrow x = y & x(\to^*_{\mathtt{LN}})^{\mathcal{A}} y \Leftrightarrow x \geq y
\end{array}$$

We obtain surjective homomorphisms (for $\mathtt{N}$ and $\mathtt{LN}$). For instance, we have:

$$\begin{array}{rclrcll}
[\mathsf{Z}]_{\mathcal{A}} & = & 0 & [\mathtt{head}(\mathtt{nil})]_{\mathcal{A}} & = & 1 & \textit{for sort } \mathtt{N} \\
[\mathtt{nil}]_{\mathcal{A}} & = & -1 & [\mathtt{cons}(\mathsf{Z}, \mathtt{nil})]_{\mathcal{A}} & = & 0 & \textit{for sort } \mathtt{LN}
\end{array}$$

7. Refutation Witnesses

A main purpose of a database language is providing primitive operations to *retrieve* information from the database satisfying some conditions. As discussed in Section 4, this is related with proving existentially quantified formulas and also giving *witnesses* of such proofs, i.e., specific values (or even formulas) for which the property holds [22]. From a logical point of view, *queries* $\varphi(\vec{x})$, where $\vec{x}$ are the *free* variables in φ, can be handled as existentially closed sentences $(\exists \vec{x})\, \varphi(\vec{x})$ which are proved and for which (possibly many) witnesses $\vec{t}$ are returned to the user. Each $\vec{t}$ is a tuple of terms such that $\mathsf{Th} \vdash \varphi(\vec{t})$ holds. Theorem proving mechanisms like *resolution* [54] can be used for this purpose.

Example 49. cf. [50, page 59] *As remarked in Example 1,* Prover9 *obtains a proof of* (2)*, which can be viewed as a* query *to the database. The proof is obtained by resolution. From the proof report, a witness* $y = \mathtt{p}$ *can be obtained.*

In logic, a *witness* for an existentially quantified sentence $(\exists x)\varphi(x)$ is a specific value b to be substituted by x in $\varphi(x)$ so that $\varphi(b)$ is true (see, e.g., [2, page 81]). Similarly, we can think of a value b such that $\neg\varphi(b)$ holds as a witness of $(\exists x)\neg\varphi(x)$ or as a *refutation witness* for $(\forall x)\varphi(x)$; we can also think of b as a *counterexample* to $(\forall x)\varphi(x)$. Witnesses given as values b from an *interpretation domain* $\mathcal{A}$, though, can be meaningless for users familiarized with the first-order language Ω at use, but not with abstract values from $\mathcal{A}$ (often automatically synthesized by some tool). Users would be happier to get *terms* t connected to witnesses b so that $t^{\mathcal{A}} = b$. Corollary 28 permits a refutation of φ by finding a model $\mathcal{A}$ of $\neg\varphi$ to conclude that $\mathcal{I} \models \neg\varphi$. In this section, we want to obtain ground instances of φ to better understand unsatisfiability of φ.

The negation of (22), i.e., of $(Q_1 x_1 : s_1) \cdots (Q_k x_k : s_k) \bigvee_{i=1}^{m} \bigwedge_{j=1}^{n_i} L_{ij}$ is

$$(\overline{Q}_1 x_1 : s_1) \cdots (\overline{Q}_k x_k : s_k) \bigwedge_{i=1}^{m} \bigvee_{j=1}^{n_i} \neg L_{ij}(x_1, \ldots, x_k) \tag{43}$$

where $\overline{Q}_i$ is $\forall$ whenever Q_i is $\exists$ and $\overline{Q}_i$ is $\exists$ whenever Q_i is $\forall$.

Example 50. *The negations of (24) and (35) are*

$$(\forall x)(\forall n) \qquad \neg\mathsf{pow}(x, \mathsf{s}(\mathsf{s}(n)), \mathsf{s}(\mathsf{s}(\mathsf{s}(\mathsf{0})))) \tag{44}$$

$$(\exists x : \mathtt{N})(\exists y : \mathtt{N})(\forall z : \mathtt{N}) \qquad \neg(\mathtt{add}(x, y) \to^* z) \vee \neg(\mathtt{add}(y, x) \to^* z) \tag{45}$$

We assume $\eta \leq k$ universal quantifiers in (43) with indices $U = \{\upsilon_1, \ldots, \upsilon_\eta\} \subseteq \{1, \ldots, k\}$ and hence $k-\eta$ existential quantifiers with indices $E = \{\epsilon_1, \ldots, \epsilon_{k-\eta}\} = \{1, \ldots, k\} - U$. In the following $\overline{\eta}$ denotes $k - \eta$. For each $\epsilon \in E$, we let $U_\epsilon = \{\upsilon \in U \mid \upsilon < \epsilon\}$ be the (possibly empty) set of indices of universally quantified variables in (43) occurring before x_ϵ in the quantification prefix of (43). Let $\eta_\epsilon = |U_\epsilon|$. Note that $U_{\epsilon_1} \subseteq U_{\epsilon_2} \subseteq \cdots \subseteq U_{\epsilon_{\overline{\eta}}}$. Let $U_\exists$ be the set of indices of universally quantified variables occurring *before* some existentially quantified variable in the quantification prefix of (43). Note that $U_\exists$

is empty whenever $\upsilon_1 > \epsilon_{k-\eta}$ (no existential quantification after a universal quantification); otherwise, $U_\exists = \{\upsilon_1, \ldots, \upsilon_\exists\}$ for some $\upsilon_\exists \leq \upsilon_\eta$. Accordingly, $U_\forall = U - U_\exists = \{\epsilon_{\overline{\eta}} + 1, \ldots, k\}$ is the set of indices of universally quantified variables occurring *after* all existentially quantified variables in the quantification prefix of (43). Note that $U_\forall$ is empty whenever $\epsilon_1 > \upsilon_\eta$ (no universal quantification after an existential quantification).

Example 51. *For the sentences in Example 50, we have the following:*

- *For (44),* $\eta = 2$, $U = \{1, 2\}$ *and* $E = \emptyset$. *Also,* $U_\exists = \emptyset$ *and* $U_\forall = \{1, 2\}$.
- *For (45),* $\eta = 1$, $U = \{3\}$ *and* $E = \{1, 2\}$. *Also,* $U_1 = U_2 = \emptyset$, $U_\exists = \emptyset$, *and* $U_\forall = \{3\}$.

Most theorem provers transform sentences into universally quantified formulas by Skolemization (see, e.g., [28]). Thus, if (43) contains existential quantifiers, we introduce new *Skolem function symbols* $sk_\epsilon : w_\epsilon \to s_\epsilon$ for each $\epsilon \in E$, where w_ϵ is the (possibly empty) sequence of η_ϵ sorts indexed by U_ϵ. Note that sk_ϵ is a *constant* if $\eta_\epsilon = 0$. The *Skolem normal form* of (43) is

$$(\forall x_{\upsilon_1} : s_{\upsilon_1}) \cdots (\forall x_{\upsilon_\eta} : s_{\upsilon_\eta}) \bigwedge_{i=1}^{m} \bigvee_{j=1}^{n_i} \neg L_{ij}(e_1, \ldots, e_k) \tag{46}$$

where for all $1 \leq q \leq k$, (i) $e_q \equiv x_q$ if $q \in U$ and (ii) $e_q \equiv sk_q(\vec{x}_{\eta_q})$ if $q \in E$, where $\vec{x}_{\eta_q}$ is the sequence of variables $x_{\nu_1}, \ldots, x_{\nu_{\eta_q}}$.

Example 52. *The Skolem normal forms of (44) and (45) are*

$$(\forall x)(\forall n) \qquad \neg \mathsf{pow}(x, \mathsf{s}(\mathsf{s}(n)), \mathsf{s}(\mathsf{s}(\mathsf{s}(\mathsf{0})))) \tag{47}$$

$$(\forall z : \mathtt{N}) \qquad \neg(\mathsf{add}(sk_x, sk_y) \to^* z) \vee \neg(\mathsf{add}(sk_y, sk_x) \to^* z) \tag{48}$$

where sk_x *and* sk_y *are constants of sort* $\mathtt{N}$.

If $E \neq \emptyset$ (i.e., (43) and (46) differ), then (46) is a sentence of an *extended* signature $\Omega^{sk} = (S, \Sigma^{sk}, \Pi)$ where Σ^{sk} extends Σ with skolem functions. Since (46) logically implies (43) [2, Section 19.2], every model $\mathcal{A}$ of (46) satisfies (43).

Definition 53 (Set of refutation witnesses). *Let* $\mathcal{A}$ *be an* Ω^{sk}*-structure with* h_{s_q} surjective *for all* $q \in U_\exists \cup E$. *The* Ω^{sk}*-sentence (46) is given a* set of refutation witnesses Φ *consisting of* Ω*-sentences* ϕ_α *for each valuation* α *of variables* $x_{\upsilon_1}, \ldots, x_{\upsilon_\exists}$ *indexed by* $U_\exists$*; each* ϕ_α *is defined as follows:*

$$(\forall x_{\epsilon_{\overline{\eta}}+1} : s_{\epsilon_{\overline{\eta}}+1}) \cdots (\forall x_k : s_k) \bigwedge_{i=1}^{m} \bigvee_{j=1}^{n_i} \neg L_{ij}(e'_1, \ldots, e'_k) \tag{49}$$

where for all $1 \leq q \leq k$, *(i)* $e'_q \equiv x_q$ *if* $q \in U_\forall$ *and (ii)* $e'_q \equiv t$ *if* $q \in U_\exists \cup E$ *and* $t \in \mathcal{T}_{\Sigma s_q}$ *is such that* $t^{\mathcal{A}} = [e_q]^{\mathcal{A}}_\alpha$.

In Definition 53 we could *fail* to find terms $t \in \mathcal{T}_{\Sigma s_q}$ if h_{s_q} is *not* surjective. Note also that, whenever E is empty, Φ is a singleton consisting of (49) which coincides with (46). Refutation witnesses are built from symbols in the original signature Ω only. They provide more intuitive *counterexamples* to property φ.

Example 54. *For (47), we obtain a single refutation witness:*

$$(\forall x)(\forall n) \qquad \neg\mathsf{pow}(x, \mathsf{s}(\mathsf{s}(n)), \mathsf{s}(\mathsf{s}(\mathsf{s}(\mathsf{0})))) \tag{50}$$

expressing that for every (natural) number there is no $n+2$-nth root.

Example 55. *For (48), the model $\mathcal{A}$ computed by* AGES *in Example 48 also includes the following interpretation for sk_x and sk_y:*

$$sk_x = 1 \quad \textit{and} \quad sk_y = 0$$

Since $U_\exists = \emptyset$ there is a single (empty) valuation $\alpha : U_\exists \to \mathcal{A}_\mathtt{N}$ which we use to obtain the refutation witnesses. Examples of terms $t \in \mathcal{T}_{\Sigma\mathtt{N}}$ such that $[t]_\mathcal{A} = [sk_x]_\mathcal{A} = 1$ are $\mathsf{head}(\mathsf{nil})$, $\mathsf{head}(\mathsf{cons}(\mathsf{head}(\mathsf{nil}), xs))$ for all ground terms xs of sort $\mathtt{LN}$, etc. For terms $t' = \mathsf{suc}^n(\mathsf{0})$ for some $n > 0$, we have $[t']_\mathcal{A} = [sk_y]_\mathcal{A} = 0$. In this way, we obtain infinitely many refutation witnesses:

$$\begin{array}{ll}
(\forall z : \mathtt{N}) & \neg(\mathsf{add}(\mathsf{head}(\mathsf{nil}), \mathsf{0}) \to^* z) \vee \neg(\mathsf{add}(\mathsf{0}, \mathsf{head}(\mathsf{nil})) \to^* z) \\
(\forall z : \mathtt{N}) & \neg(\mathsf{add}(\mathsf{head}(\mathsf{nil}), \mathsf{suc}(\mathsf{0})) \to^* z) \vee \neg(\mathsf{add}(\mathsf{suc}(\mathsf{0}), \mathsf{head}(\mathsf{nil})) \to^* z) \\
& \vdots
\end{array}$$

These refutation witnesses express that whenever $\mathsf{head}(\mathsf{nil})$ is an argument of add, the normal form of the addition depends on the argument that holds it.

Proposition 56. *For every Ω^{sk}-structure $\mathcal{A}$, $\mathcal{A} \models (46)$ if and only if $\mathcal{A} \models \Phi$.*

Proof. For the *only if* part, consider an arbitrary valuation α of variables $x_{v_1}, \ldots, x_{v_\exists}$. Since $\mathcal{A} \models (46)$, we have that

$$\left[(\forall x_{\epsilon_{\overline{\eta}}+1} : s_{\epsilon_{\overline{\eta}}+1}) \cdots (\forall x_k : s_k) \bigwedge_{i=1}^{m} \bigvee_{j=1}^{n_i} \neg L_{ij}(e_1, \ldots, e_k) \right]_\alpha^\mathcal{A} \tag{51}$$

is true. Now, consider $\phi_\alpha \in \Phi$. For all $1 \leq p \leq k$, we have $[e'_p]_\alpha^\mathcal{A} = [e_p]_\alpha^\mathcal{A}$. Thus, $\mathcal{A} \models \phi_\alpha$ and therefore $\mathcal{A} \models \Phi$. With regard to the *if* part, we have to prove that every valuation α for $x_{v_1}, \ldots, x_{v_\eta}$ satisfies the *matrix* formula of (46), i.e., $[\bigwedge_{i=1}^m \bigvee_{j=1}^{n_i} \neg L_{ij}(e_1, \ldots, e_k)]_\alpha^\mathcal{A}$ is true. Consider the formula $\phi_\alpha \in \Phi$ for α (restricted to $x_{v_1}, \ldots, x_{v_\exists}$), i.e.,

$$(\forall x_{\epsilon_{\overline{\eta}}+1} : s_{\epsilon_{\overline{\eta}}+1}) \cdots (\forall x_k : s_k) \bigwedge_{i=1}^{m} \bigvee_{j=1}^{n_i} \neg L_{ij}(e'_1, \ldots, e'_k)$$

Since $\mathcal{A} \models \phi_\alpha$, we know that $[\bigwedge_{i=1}^m \bigvee_{j=1}^{n_i} \neg L_{ij}(e'_1, \ldots, e'_k)]_\alpha^\mathcal{A}$ is true. By definition of ϕ_α, $[e_p]_\alpha^\mathcal{A} = [e'_p]_\alpha^\mathcal{A}$ for all $1 \leq p \leq k$. We conclude that $\mathcal{A} \models (46)$ holds. □

Proposition 57. *Let Ω be a signature, Th be a theory, φ be a sentence (22), and $\mathcal{A}$ be a model of Th such that for all negative literals $L_{ij} = \neg P(\vec{t})$ with $P \in \Pi_w$ and substitutions σ, if $h(\sigma(\vec{t})) \in P^{\mathcal{A}}$ then $\sigma(\vec{t}) \in P^{\mathcal{I}}$. For all $\phi \in \Phi$, $\mathcal{I} \models \phi$.*

Proof. The refutation witnesses ϕ for (the Skolem normal form of) $\neg(22)$ are of the form $(\forall x_{\epsilon_{\overline{\eta}}+1} : s_{\epsilon_{\overline{\eta}}+1}) \cdots (\forall x_k : s_k) \bigwedge_{i=1}^{m} \bigvee_{j=1}^{n_i} \neg L_{ij}(t_1, \ldots, t_{\epsilon_{\overline{\eta}}}, x_{\epsilon_{\overline{\eta}}+1}, \ldots, x_k)$. If $\mathcal{I} \models \phi$ does not hold, then we have $\mathcal{I} \models \neg\phi$ instead, i.e., there are terms $t_{\epsilon_{\overline{\eta}}+1}, \ldots, t_k$ of sorts $s_{\epsilon_{\overline{\eta}}+1}, \ldots, s_k$, respectively, such that

$$\mathcal{I} \models \bigwedge_{j=1}^{n_i} L_{ij}(t_1, \ldots, t_{\epsilon_{\overline{\eta}}}, t_{\epsilon_{\overline{\eta}}+1}, \ldots, t_k)$$

for some $1 \leq i \leq m$. Positive literals L_{ij}, i.e., atoms, are preserved under homomorphism, and for all $L_{ij}(t_1, \ldots, t_{\epsilon_{\overline{\eta}}}, t_{\epsilon_{\overline{\eta}}+1}, \ldots, t_k) = \neg P(t_1, \ldots, t_{\epsilon_{\overline{\eta}}}, t_{\epsilon_{\overline{\eta}}+1}, \ldots, t_k)$, we must have $\mathcal{A} \models L_{ij}(t_1, \ldots, t_{\epsilon_{\overline{\eta}}}, t_{\epsilon_{\overline{\eta}}+1}, \ldots, t_k)$; otherwise, since $\mathcal{I} \models L_{ij}$ holds, i.e., $\vec{t} \notin P^{\mathcal{I}}$, we contradict our initial assumption for negative literals. Therefore,

$$\mathcal{A} \models \bigvee_{i=1}^{m} \bigwedge_{j=1}^{n_i} L_{ij}(t_1, \ldots, t_{\epsilon_{\overline{\eta}}}, t_{\epsilon_{\overline{\eta}}+1}, \ldots, t_k).$$

Thus, for the valuation α of variables $x_{\epsilon_{\overline{\eta}}+1}, \ldots, x_k$ given by $\alpha(x_p) = t_p^{\mathcal{A}}$ for all $\epsilon_{\overline{\eta}} + 1 \leq p \leq k$, we have that

$$\left[\bigvee_{i=1}^{m} \bigwedge_{j=1}^{n_i} L_{ij}(t_1, \ldots, t_{\epsilon_{\overline{\eta}}}, x_{\epsilon_{\overline{\eta}}+1}, \ldots, x_k) \right]_{\alpha}^{\mathcal{A}}$$

is true, contradicting $\mathcal{A} \models \phi$, which holds by definition of Φ and Proposition 56. □

Corollary 58. *If (22) is positive, then for all $\phi \in \Phi$, $\mathcal{I} \models \phi$.*

8. Some explored applications

Some applications of the theory described in the previous sections have been explored for rewriting-based systems. In this section we give a brief account. In order to save space, we point to appropriate references for additional examples. Figures 4–6 show some properties of interest about (unsorted) CTRSs which have been investigated in the literature. Here, x, y, z denote variables, s and t denote *ground terms*, $s_1, \ldots, s_n$, $t_1, \ldots, t_n$ and u are arbitrary terms with $\vec{x}$ referring all variables in such terms.

Property	φ	Negation
*Reach*able	$s \to^* t$	Unreachable
*Feas*ible	$(\exists \vec{x}) s_1 \to^* t_1 \wedge \cdots \wedge s_n \to^* t_n$	Infeasible
*Join*able	$(\exists x)\ (s \to^* x \wedge t \to^* x)$	Non-joinable
*Meet*able	$(\exists x)\ (x \to^* s \wedge x \to^* t)$	
Peak	$(\exists x)\ (x \to s \wedge x \to t)$	
*Red*ucible	$(\exists x)\ t \to x$	Irreducible
*Narr*owable	$(\exists \vec{x}, y)\ u \to y$	nonvariable reduction!
*Cycl*ing *t*erm	$(\exists x)\ t \to x \wedge x \to^* t$	
*Cycl*ing system	$(\exists x, y)\ x \to y \wedge y \to^* x$	

Figure 4: Feasibility problems for CTRSs

8.1. (In)feasibility problems

Figure 4 shows some examples of *feasibility* properties for CTRSs, where given pairs of terms s_i, t_i, $1 \leq i \leq n$, we want to find a substitution σ such that, for all $1 \leq i \leq n$, $\sigma(s_i) \bowtie \sigma(t_i)$ holds (with $\bowtie$ either $\to$ or $\to^*$). From a logical point of view, they can be handled as ECBCAs φ and then *proved* by using Corollary 14 and *disproved* by using Corollary 47 with $\mathsf{H}_\varphi = \mathsf{N}_\varphi = \emptyset$. This approach has been implemented in the tool infChecker [24] using Prover9 as a backend to use Corollary 14 and Mace4 and AGES to implement the use of Corollary 47. infChecker proved to be the most powerful tool among the participants in the new *infeasibility category* of the 2019 Confluence Competition [44, Section 5.2].[11] Quite surprisingly, with our logic-based, generic techniques, we could handle most examples coming from papers developing several specific techniques to deal with these problems (including tree automata [43], unification [59], theorem proving [58] and interpretation methods [1]).

8.2. Other properties of rewriting-based systems expressed as ECBCAs

Figure 5 shows other properties of rewriting-based systems which can also be expressed as ECBCAs. In this case, though, further predicate symbols, appropriately defined by a first-order theory, are used. We discuss them briefly:

Root reduction. A predicate $\xrightarrow{\Lambda}$ is used to represent *root-reduction*, i.e., rewriting at the root of terms only. Its Horn theory $H_{\xrightarrow{\Lambda}_{\mathcal{R}}}$, where reductions with $\xrightarrow{\Lambda}_{\mathcal{R}}$ are *not* propagated below the root of terms (as done by rules $C_{f,i}$ in Figure 2), is as follows: for each rule $\ell \to r \Leftarrow s_1 \to t_1, \ldots, s_n \to t_n$, we have a sentence:

$$(\forall x_1, \ldots, x_k)\ s_1 \to^* t_1 \wedge \cdots \wedge s_n \to^* t_n \Rightarrow \ell \quad \xrightarrow{\Lambda} \quad r \tag{52}$$

[11] Competition results here: `http://cops.uibk.ac.at/results/?y=2019&c=INF`. infChecker also implements some transformations for CTRSs [36, 24], but their contribution to the results in the competition is small. The main bulk corresponds to Corollaries 14 and 47.

Property	φ	Negation
Root reducible	$(\exists x)\ t \xrightarrow{\Lambda} x$	
*Conv*ertible	$s \to t \vee t \to s$	
*Loop*ing *t*erm	$(\exists x, y)\ t \to x \wedge x \to^* y \wedge y \trianglerighteq t$	
*Loop*ing system	$(\exists x, y, z)\ x \to y \wedge y \to^* z \wedge z \trianglerighteq x$	
Redex-reducible	$(\exists \vec{x})\ \bigvee_{\ell \to r \in \mathcal{R}} t \to^* \ell$	Head-normal form
*C*ond. *Redex*-reducible	$(\exists \vec{x}, y)\ \bigvee_{\ell \to r \Leftarrow c \in \mathcal{R}} (t \to^* \ell \wedge \ell \xrightarrow{\Lambda} y)$	Head-normal form

Figure 5: ECBCA properties about rewriting-based systems

in $H_{\xrightarrow{\Lambda}_{\mathcal{R}}}$ (where $x_1, \ldots, x_k$ are the variables occurring in the rule) and nothing else. The conditions in the rules are evaluated with $\to^*_{\mathcal{R}}$ rather than with $\xrightarrow{\Lambda}{}^*_{\mathcal{R}}$. For this reason, no definition of the reflexive and transitive closure of $\xrightarrow{\Lambda}$ is given.

Subterm. Predicate symbol $\trianglerighteq$ represents the *subterm relation* among terms. The corresponding Horn theory $H_{\trianglerighteq}$ is:

$$\begin{array}{rcll} (\forall x)\ x & \trianglerighteq & x & (\forall x_1, \ldots, x_k)\ f(x_1, \ldots, x_k) \trianglerighteq x_i \\ (\forall x, y, z)\ x \trianglerighteq y \wedge y \trianglerighteq z \Rightarrow x & \trianglerighteq & z & \text{for each } f \in \mathcal{F} \text{ and } 1 \leq i \leq k \end{array}$$

Examples of the use of our techniques for proving properties in Figure 5 can be found in [33, Sections 5.4 and 5.5].

8.3. More general properties of (C)TRSs

A TRS $\mathcal{R}$ is *top-terminating* if no infinitary reduction sequence performs infinitely many root-rewritings [13]. Top-termination is important in the semantic description of lazy languages as it is an important ingredient to guarantee that every initial expression has an infinite normal form [13, 15]. Accordingly, given a *dummy* sort S, the *negation* of

$$(\exists x : \mathrm{S})(\forall n \in \mathbb{N})(\exists y : \mathtt{S})\ x(\to^* \circ \xrightarrow{\Lambda})^n y \tag{53}$$

(which claims for the existence of a term with infinitely many rewriting steps at top) captures top-termination. We introduce a new predicate $\to_{\star,\Lambda}$ for the composition $\to^* \circ \xrightarrow{\Lambda}$ of the many-step rewriting relation $\to^*$ and topmost rewriting $\xrightarrow{\Lambda}$ in Section 8.2. Sequences $s \to^n_{\star,\Lambda} t$ meaning that s $\to_{\star,\Lambda}$-reduces into t in $n+1$ $\to_{\star,\Lambda}$-steps are defined as follows:

$$(\forall x, y, z : \mathrm{S}) \qquad x \to^* y \wedge y \xrightarrow{\Lambda} z \Rightarrow x \to^0_{\star,\Lambda} z \tag{54}$$

$$(\forall x, y, z : \mathtt{S})(\forall n \in \mathbb{N}) \qquad x \to^0_{\star,\Lambda} y \wedge y \to^n_{\star,\Lambda} z \Rightarrow x \to^{n+1}_{\star,\Lambda} z \tag{55}$$

Overall, the sentence φ to be disproved is:

$$(\exists x : \mathtt{S})(\forall n : \mathit{Nat})(\exists y : \mathtt{S})\ x \to^n_{\star,\Lambda} y \tag{56}$$

where *Nat* is intended to be interpreted as $\mathbb{N}$. Figure 6 shows further well-known properties of CTRSs. Examples of use can be found in [34, Section 5.2].

Property	φ
*G*round *red*ucible	$(\forall \vec{x})\ (\exists y)\ t(\vec{x}) \to y$
*Comp*letely defined f	$(\forall \vec{x})(\exists y)\ f(x_1, \ldots, x_k) \to y$
*Comp*letely defined TRS	$(\forall \vec{x})(\exists y_1, \ldots, y_D) \bigwedge_{i=1}^{D}\ f_i(x_1, \ldots, x_{ar(f_i)}) \to y_i$ for D the number of nonconstant defined symbols
*Prod*uctive	$(\forall x)(\exists \vec{y})\ \bigvee_{c \in \mathcal{C}} x \to^* c(y_1, \ldots, y_k)$
*Inf*initely root-reducible	$(\exists x)(\forall n \in \mathbb{N})(\exists y)\ x(\to^* \circ \overset{\Lambda}{\to})^n y$
*Norm*alizing term	$(\exists x)\ (s \to^* x \wedge \neg(\exists y)\ x \to y)$
*Norm*alizing TRS (WN)	$(\forall x)(\exists y)\ (x \to^* y \wedge \neg(\exists z)\ y \to z)$
Locally confluent (WCR)	$(\forall x, y, z)\ x \to y \wedge x \to z \Rightarrow (\exists u)\ x \to^* u \wedge z \to^* u$
*Conf*luent (CR)	$(\forall x, y, z)\ x \to^* y \wedge x \to^* z \Rightarrow (\exists u)\ x \to^* u \wedge z \to^* u$

Figure 6: Arbitrary properties about rewriting-based systems

9. Related work

9.1. Reiter's assumptions for relational databases

In [51], as part of his logical approach to relational databases, Reiter defines a *Domain Closure Axiom* (DCA, "the individuals occurring in the database are all and only the existing individuals") and a *Unique Name Axiom* (UNA, "individuals with distinct names are distinct")[12] which correspond to (31) and (32), respectively [52, page 207]. Reiter's purpose is making satisfiability of formulas in $\mathcal{I}_{DB}$ *equivalent* to provability in a new theory including DB, DCA, UNA and further axioms (see [52, Section 3.1]).

Our results provide an alternative approach: since relational data bases can be seen as Horn theories DB, we can prove sentences φ by using theorem proving with respect to DB (Corollary 14); if this fails, disprove φ by using Corollary 47 with DB, and H_φ and N_φ appropriately defined: since relational databases use finitely many constant symbols only,

- $\mathsf{H}_\varphi = \cup_{s \in U} \mathsf{SuH}_s^{T_s}$ for U the set of sorts with variables universally quantified in φ and T_s the set of constants of sort s in the signature.
- N_φ is easily computable if φ contains negative literals.

9.2. Dealing with negative literals in deductive databases and logic programming

Theories obtained from Horn clauses can be used to prove positive literals only (Remark 15). In logic programming, negative literals are handled by adopting Reiter's Closed World Assumption (CWA): everything which cannot be proved is assumed to be *false* [50]. This is consistent with the model-theoretic semantics of logic programs $\mathcal{P}$ where atoms not belonging to $\mathcal{I}_\mathcal{P}$ are considered

[12] The description of the axioms is taken from [52, page 191], where they are called *assumptions*. Furthermore, in [51, page 247], UNA is actually introduced as *E-saturation*.

false. Clark proved that this view of negation is consistent with provability of negative literals (in the usual sense) with respect to a *completed* theory $Comp(\mathcal{P})$ associated to program $\mathcal{P}$ [6]. Clark's approach to prove negative literals $\neg A$ by theorem proving (i.e., $Comp(\mathcal{P}) \vdash \neg A$) may fail to work.

Example 59. *The following logic program $\mathcal{P}$ [19, Example 6.1]:*

```
edge(a,b). edge(c,d). edge(d,c).
reachable(a).
reachable(X) <- edge(Y,X), reachable(Y).
```

represents the graph:

and predicate `reachable` *for vertices reachable from* `a`*. Gelfond and Lifschitz note that* $\neg$`reachable(c)` *cannot be proved from Clark's completion of* `reachable`*:*

$$\texttt{reachable}(x) \quad \Leftrightarrow \quad (x = \texttt{a} \vee (\exists y)\, \texttt{reachable}(y) \wedge \texttt{edge}(y, x)) \qquad (57)$$

This is not surprising: Clark's completion precisely captures proofs by SLDNF resolution and $\neg$`reachable(c)` *cannot be proved as a failure of* `reachable(c)` *by SLDNF resolution due to an infinite computation branch.*

We prove that $\neg$`reachable(c)` *is a semantic property of $\mathcal{P}$ by finding a model $\mathcal{A}$ of $\mathcal{P} \cup \{\neg$*`reachable(c)`*$\}$ (note that* `reachable(c)` *is an ECBCA) with* Mace4*: the domain is $\mathcal{A} = \{0, 1\}$; constants are interpreted by $\texttt{a}^{\mathcal{A}} = \texttt{b}^{\mathcal{A}} = 0$, $\texttt{c}^{\mathcal{A}} = \texttt{d}^{\mathcal{A}} = 1$; for predicate symbols: $\texttt{reachable}^{\mathcal{A}}(x) \Leftrightarrow x = 0$ and $\texttt{edge}^{\mathcal{A}}(x, y) \Leftrightarrow x = y$.*

The example shows that approaching negation by satisfiability in arbitrary structures can be useful when negation-as-failure does not work.

9.3. Entailment in ASP

Stable models X for ASP programs $\mathcal{P}$ are often obtained as least Herbrand models $\mathcal{I}_{\mathcal{P}^X}$ of basic logic programs $\mathcal{P}^X$, often called *reducts* so that the (fixpoint) *stability* condition $\mathcal{I}_{\mathcal{P}^X} = X$ is satisfied. For NLPs $\mathcal{P}$, reducts are obtained by first considering all ground instances of the rules in $\mathcal{P}$ and (i) removing rules containing (extended) literals *not* B with $B \in X$, and then (ii) removing negative literals from any other rule [19, Section 2].

Example 60. *(cont. Example 23) For $\mathcal{H} = \{\texttt{p(a)}\}$ in Example 8, $\mathcal{P}^{\mathcal{H}} = \{\texttt{p(a)}\}$. Since $\mathcal{I}_{\mathcal{P}^{\mathcal{H}}} = \mathcal{H}$, it follows that $\mathcal{H}$ is stable. For $\mathcal{H}' = \{\texttt{p(b)}\}$, we have $\mathcal{P}^{\mathcal{H}'} = \{\texttt{p(b)}\}$, i.e., $\mathcal{H}'$ is a stable model of $\mathcal{P}$.*

An ASP program $\mathcal{P}$ *entails* a ground literal L (written $\mathcal{P} \models L$) if L is satisfied by *every* answer set X of $\mathcal{P}$. ASP queries q are solved by *entailment*: "*yes*", if $\mathcal{P} \models q$; "*no*", if $\mathcal{P} \models \neg q$, and "*unknown*" otherwise [18, Defs. 7.2.4 and 7.2.5].

Remark 61 (Proving entailment with NLPs). *For NLP programs $\mathcal{P}$, we have $\mathcal{P} \models L$ iff for all stable models X of $\mathcal{P}$, $\mathcal{I}_{\mathcal{P}^X} \models L$. Therefore, we can prove the entailment of a sentence φ by an NLP program $\mathcal{P}$ if for all answer sets X of $\mathcal{P}$, φ is a semantic property of the reduct $\mathcal{P}^X$ (a set of ground Horn clauses).*

In this way, the techniques discussed in this paper could also be of some use in ASP programming.

Example 62. *For $\mathcal{P}$ in Example 23, we have*

$$\begin{array}{rcl} \mathcal{P} \models \mathtt{p(a)} & \Leftrightarrow & \mathcal{I}_{\mathcal{P}^{\mathcal{H}}} \models \mathtt{p(a)} \text{ and } \mathcal{I}_{\mathcal{P}^{\mathcal{H}'}} \models \mathtt{p(a)} \\ & \Leftrightarrow & \text{true and false} \\ & \Leftrightarrow & \text{false} \end{array}$$

and similarly for $\neg\mathtt{p(a)}$. The answer to queries $\mathtt{p(a)}$ and $\neg\mathtt{p(a)}$ is "unknown".

Although reducts are usually obtained after grounding the original ASP program, limiting the attention to constant functions only is frequent in ASP systems. Thus, reducts $\mathcal{P}^X$ representing stable sets X can be computed and the application of our results as explained in Remark 61 would also be feasible.

9.4. The First-Order Theory of Rewriting

The so-called *first-order theory of rewriting* for a TRS $\mathcal{R}$ (*FOThR* in the following) uses a restricted first-order language *without* constant or function symbols, and with only two predicate symbols $\to$ and $\to^*$. Predicate symbols are *interpreted* on an *intended model* which gives meaning to $\to$ and $\to^*$ as the one-step and many-step rewrite relations $\to_{\mathcal{R}}$ and $\to^*_{\mathcal{R}}$ for $\mathcal{R}$ on *ground terms*, respectively [12]. This is just the least Herbrand model $\mathcal{I}_{\mathcal{R}}$ for the Horn theory $\overline{\mathcal{R}}$ of $\mathcal{R}$ (see Section 4.5). *FOThR* is often used to express and verify properties of TRSs. For instance, confluence or local confluence (see Figure 6) fit this format. Given a TRS $\mathcal{R}$ and a formula φ in the language of *FOThR*, $\mathcal{I}_{\mathcal{R}} \models \varphi$ (i.e., the satisfiability of φ in $\mathcal{I}_{\mathcal{R}}$) actually *means* that the property expressed by φ *holds* for $\mathcal{R}$. For instance $\mathcal{I}_{\mathcal{R}} \models CR$ means '$\mathcal{R}$ is ground confluent'. And $\neg(\mathcal{I}_{\mathcal{R}} \models CR)$, which is equivalent to $\mathcal{I}_{\mathcal{R}} \models \neg CR$ means '$\mathcal{R}$ is *not* ground confluent'. Decision algorithms for these properties exist for restricted classes of TRSs $\mathcal{R}$ like left-linear right-ground TRSs, where variables are allowed in the left-hand side of rules (with no repeated occurrences of the same variable) but disallowed in the right-hand side [49]. However, a simple fragment of *FOThR* like the *First-Order Theory of One-Step Rewriting*, where only predicate $\to$ is allowed, has been proved undecidable even for *linear* TRSs [62]. Some specialized approaches to deal with reachability analyses in proofs of confluence or termination of rewriting have been recently proposed [60].

Other approaches like the ITP tool, *a theorem prover that can be used to prove properties of membership equational specifications* [9] work similarly: the tool can be used to verify such properties *with respect to ITP-models* which are actually special versions of the Herbrand model of the underlying theory. Then, one may have similar decidability problems as discussed for *FOThR*.

In contrast to *FOThR*, we use the full expressive power of FOL to represent sophisticated rewrite theories where sorts, conditional rules and equations, membership predicates, etc., are allowed. We do not impose any restriction on the class of rewrite systems we can deal with. In contrast to *FOThR*, where function symbols are disallowed in formulas, we use sentences involving arbitrary terms. Also in contrast to *FOThR*, with a single allowed model $\mathcal{H}_{\mathcal{R}}$, we permit the *arbitrary interpretation* of the underlying first-order logic language for proving properties. The application of this approach to well-known problems in rewriting leads to new methods which show their practical usefulness (see Section 8.1). In *FOThR*, strong restrictions are imposed on the shape of formulas that can be treated. For instance, most sentences in Figures 4–6 are not expressible in *FOThR*, as they involve specific *terms* with or without variables.

9.5. Verification of safety properties of rewrite systems

In [29], Lisitsa investigates verification of safety properties of TRSs $\mathcal{R}$. He uses first-order descriptions $\Psi_{\mathcal{R}}$ of TRSs $\mathcal{R}$. $\Phi_{\mathcal{R}}$ is analogous to $\overline{\mathcal{R}}$ as given in Section 4.5. However, only a predicate R (instead of $\rightarrow$ or $\rightarrow^*$, but which actually plays the role of $\rightarrow^*$) is used, and rewrite rules do not have conditional part, as $\mathcal{R}$ is a TRS.

Targetted safety properties are encoded as reachability problems $t_1 \rightarrow^*_{\mathcal{R}} t_2$ where t_1 and t_2 are *ground* terms respectively belonging to the language of appropriate tree automata $\mathcal{A}_I$ and $\mathcal{A}_U$ over the signature $\mathcal{F}$ extended with appropriate finite sets of constans Q_I and Q_U respectively denoting the sets of states of the automata. Without loss of generality, $\mathcal{F}$, Q_I, and Q_U are assumed to be disjoint. The rewriting steps are issued using $\mathcal{R}$. Any tree automaton $\mathcal{A}$ can be simulated as a TRS (also denoted $\mathcal{A}$) and the language $\mathcal{L}(\mathcal{A})$ accepted by the automaton can be described by means of a set of reachability problems for the TRS $\mathcal{A}$. Overall, the considered safety problems are rephrased as the formula

$$\psi_P = (\exists x)(\exists y) \bigvee_{q_i \in Q_I, q_u \in Q_U} R(x, q_i) \wedge R(x, y) \wedge R(y, q_u) \tag{58}$$

The *refutation* of ψ_P proceeds by finding a model $\mathcal{A}$ of $\Phi_P \cup \{\neg\psi_P\}$, where Φ_P is the union of $\Phi_{\mathcal{A}_I}$, $\Phi_{\mathcal{A}_U}$ and $\Phi_{\mathcal{R}}$ [29, Theorem 3.5]. Since Φ_P is a first-order theory and ψ_P is an ECBCA, this is a particular case of Corollary 28. This is also true for the variants of the aforementioned safety problem for TRSs investigated by Lisitsa in [29, Sections 3.6 and 4].

9.6. Term rewriting systems as logic programs

Gallagher and Rosendahl use Horn clauses to encode TRSs and then investigate reachability issues [17]. Their approach, however, is quite different from ours. As explained in Section 4.5, our starting point is not just a TRS $\mathcal{R}$ but the *inference rules* $\mathfrak{I}(\mathcal{R})$ associated to $\mathcal{R}$ in a given computational logic which can be different for the *same* TRS (e.g., a TRS $\mathcal{R}$ yields a different inference system and logical theory when considered as a context-sensitive rewrite system [32]).

In this way, we make explicit not only the rules of the TRS but also the description of the considered operational semantics (and possibly other relations). Such a semantic description is implicit (and therefore *fixed*) in their encoding. Also, we view rewrite rules $\ell \to r$ just as *atoms* for a binary predicate $\to$, whereas they translate each rewrite rule into several Horn clauses which flatten the original terms to simulate pattern matching [17, Section 3]. Furthermore, we do not restrict the attention to TRSs; our method applies to more general rewriting-based systems.

9.7. Verification of concurrent systems

In [30], Lisitsa investigates reachability problems for transition systems represented as finite automata (with the possibility of using some arithmetic operators together with standard arithmetic properties encoded as equations) by using a logic-based approach similar to ours. Only monadic predicates are considered. A reachability predicate *Reach* is intended to hold if a given state is reachable from the initial state of the considered automaton. Furthermore, only ECBCA sentences are considered to express semantic properties.

9.8. Protocol verification

In [56], Selinger shows how to encode cryptographic protocols as first-order formulas so that a proof of correctness of a given protocol can be pursued by just finding a model of a set of axioms representing properties of cryptographic properties (see [56, Figure 1]) together with a description of some particular protocol, and (the negation of) a formula Ψ which represents a violation of secrecy (and thus, its negation represents the desired correctness requirement of the protocol), see [56, page 79]. In order to illustrate his technique, Selinger uses a corrected version of Needham-Schroeder protocol derived from the analysis of Lowe [31], for which he obtains a proof of correctness by satisfiability [56, Section 2.5]. Goubault-Larrecq further develops Selinger's approach by showing that formally checkable proofs of correctness of protocols can be obtained from the obtained models. He also investigates the specific interest (concerning decidability issues) of using *finite* models in protocol verification (despite he shows that there are protocols requiring infinite models too). In these two papers, the considered formulas Ψ are ECBCAs. Thus, the correctness of the approach actually follows from Corollary 28. We think that more general properties of security protocols which do not fit the ECBCA format could be analyzed by using our methods.

A similar technique is also used by Jürjens and Weber [26], where theories are restricted to *limit sentences*, which are universally quantified implications where some variables in the consequent can be quantified with the *uniqueness quantifier* $\exists!$ instead[13]. Limit sentences properly extend Horn clauses. Their use is motivated by specific requirements that the computed models are intended

[13] As usual, $(\exists!\, x)P(x)$ abbreviates $(\exists x)(P(x) \wedge (\forall y)(P(y) \Rightarrow x = y))$

to fulfill when dealing with protocols involving equational components, see [26, Theorem 2]. On the other hand, sentences to be refuted (called *conjectures*) must be universally quantified conjunctions of atoms $(\forall \vec{x}) \bigwedge_{i=1}^{n} A_i$ [26, Section 2]. Also, the paper considers finite models only, although no strong technical reason is apparently given to justify this restriction (which is not imposed in [56], for instance). Jürjens and Weber's approach involves universally quantified conjunctions of atoms which are not ECBCAs. Thus, an interesting subject for future work is investigating how to exploit the specific shape of the considered theories Th (e.g., limit sentence) to guarantee soundness without adding sentences reinforcing surjectivity, as required by Corollary 28.

9.9. Proof by consistency

The idea of proving logic formulas φ with respect to a theory Th by showing that $\mathsf{Th} \cup \{\varphi\}$ is not contradictory is called *proof by consistency* by Kapur and Musser [27]. The authors point that this proof method is, in general, unsound, but can be faithfully used with *strongly complete* proof systems, which are those guaranteeing that, whenever a formula φ can be added as an axiom to the proof system without introducing inconsistencies, then φ is also a theorem of the system. The authors remark that first-order predicate calculus are *not* strongly complete. Peano arithmetics is not strongly complete either, although *when restricted to formulas which are universally quantified equations* is strongly complete [27, page 126]. Kapur and Musser focus on many-sorted *equational* proof systems consisting of a subset C of ground terms and a set of equations E where equational logic is used as proof system. Their proof by consistency method is used to prove equations (with implicit universal quantification) with respect to the considered systems. Furthermore, additional requirements like *unambiguity* are required on the considered systems (i.e., theories, see [27, Theorem 9.2], for instance. Moreover, Kapur and Musser do not use satisfiability to show consistency of the theory (although both notions are equivalent[14]). Instead, proof-theoretic (rewriting-based) methods are used to prove consistency.

9.10. Proving semantic properties by satisfiability vs. theorem proving

When proving semantic properties of a theory Th, the use of Corollary 47 may have interesting practical advantages. In principle, by Corollary 14, a property φ of Th can be verified using theorem proving. However, this is not always possible (see Example 2 regarding the failing attempts to prove (3) and $\neg$(3) as theorems of Teach). In particular, it is well-known that proving *negative* statements $\neg\varphi$, where φ is a positive formula, is not possible if (as often happens) Th is a Horn theory. For instance, although $\neg$(3) could not be proved (for Teach) with Prover9, it is proved in Example 5 by using our results together with Mace4.

[14] A theory Th is called consistent if no formula φ and its negation $\neg\varphi$ is provable [40, Page 72]. This is a proof-theoretic definition. The (equivalent) semantic version is: a theory is consistent if it has a model [25, page 41]. Since theories consist of sentences, this is equivalent to the satisfiability of each sentence of the theory.

Of course, although *disproving* positive properties φ by using our method is feasible and useful (note that $\mathsf{N}_\varphi = \emptyset$ in this case), an attempt to directly *prove* φ using Corollary 47 requires considering $\overline{\varphi} = \neg\varphi$ which would contain negative literals, thus requiring a nonempty component $\mathsf{N}_{\neg\varphi}$ which can be difficult to compute in many cases (in particular, for predicate symbols P with arguments of sorts s with infinite sets $\mathcal{T}_{\Sigma s}$ of ground terms).

10. Conclusions and future work

We have presented a semantic approach to prove semantic properties of computational systems described as a first-order theory Th. Semantic properties are defined as first-order sentences φ which are satisfiable in the interpretation $\mathcal{I}_{\mathsf{Th}}$ associated to the set of ground consequences of Th. We have proved that semantic properties φ can be proved by finding an arbitrary model $\mathcal{A}$ of $\mathsf{Th} \cup \mathsf{H} \cup \mathsf{N} \cup \{\varphi\}$, where the possibly empty theories H and N depend on φ only (Corollary 47). For Horn theories Th, theorem proving can also be used to verify (positive) semantic properties φ as theorems of Th (Corollary 14). Our approach is applicable in well-known realms like relational and deductive databases, logic programming, answer-set programming, and rewriting-based systems. Of course, the existence of (finite or infinite, although finitely presented) models for a given theory is not guaranteed. Much less for finitely presented models computed by using some *specific* approach like linear algebra (as it is the case of AGES). Our (still short) practical experience shows that the method can be competitive when compared to other approaches (see the discussion at the end of Section 8.1). This observation encourages us to further investigate the deep reasons for this, by means of theoretical and empirical studies.

Our work is also helpful to clarify the outcome of model generation tools when used together with theorem provers[15] in some applications: program analysis, solving queries, etc. Of course, the main purpose of these tools is *theorem proving* and finding a countermodel of φ, i.e., a model of $\mathsf{Th} \cup \{\neg\varphi\}$, actually shows that φ is not a theorem of Th, i.e., $\mathsf{Th} \vdash \varphi$ does not hold (this is the main use of such countermodel generation feature). Although concluding $\mathsf{Th} \vdash \neg\varphi$ from such a countermodel would, in general, be wrong (but non-expert users could be tempted to do it), our results show that, depending on the shape of φ, the conclusion $\mathcal{I}_{\mathsf{Th}} \models \neg\varphi$ (which, following Clark, in most cases is the intended meaning of $\neg\varphi$) could be faithfully affirmed, perhaps after adding sentences reinforcing surjectivity, etc.

The two main practical results presented in this paper (Corollaries 14 and 47) are the basis of the tool infChecker, which is able to prove restricted semantic properties of CTRSs (see Section 8.1). The results obtained by this tool are very encouraging: the tool won the first international competition of infeasibility

[15] for instance, Prover9/Mace4, but also Alt-Ergo (`https://alt-ergo.ocamlpro.com/`), PDL-tableau (`http://www.cs.man.ac.uk/~schmidt/pdl-tableau/`) and Princess (`http://www.philipp.ruemmer.org/princess.shtml`).

tools as part of the 2019 Confluence Competition (CoCo). The application of our techniques to other computational systems (databases, cryptographic protocols, logic and answer set programming) is therefore promising, and we plan to explore their performance in the future.

We also plan to develop a fully general implementation of our results as an extension of the tool AGES, which already provides the necessary ability to generate finite and infinite models of many-sorted first-order theories. It is, however, unable to deal with the automatic generation of theories H and N which could be required to verify a given semantic property φ, or compute refutation witnesses. This is a subject for future work. Also, our research suggests that further investigation on the generation of models for many-sorted theories that combines the use of finite and infinite domains is necessary. For instance, [35] explains how to generate such models by interpreting the sort, function, and predicate symbols by using linear algebra techniques. This is implemented in AGES. Domains are defined as the solutions of matrix inequalities, possibly restricted to an underlying set of values (e.g., $\mathbb{Z}$); thus, finite and infinite domains can be obtained as particular cases of the same technique. Since piecewise definitions are allowed, we could eventually provide fully detailed descriptions of functions and predicates by just adding more pieces to the interpretations. However, such a flexibility is expensive. In contrast, Mace4 is based on a different principle (similar to [28]) and it is really fast, but only finite domains can be generated. This is a problem, for instance, when using Proposition 46 to guarantee surjectivity of homomorphisms $h_s : \mathcal{T}_{\Sigma s} \to \mathcal{A}_s$. Even though $\mathcal{A}_s$ is finite, we still need to be able to interpret *Nat* as $\mathbb{N}$, which is not possible with Mace4. For this reason, Example 48 (where the computed structures $\mathcal{A}$ have finite domains for the 'proper' sorts `N` and `LN`, and only *Nat* is interpreted as an infinite set $\mathbb{N}$) could not be handled with Mace4, or with similar tools that are able to deal with sorts (e.g., SEM [64] or the work in [53]) but which generate finite domains only.

Acknowledgements. I thank the anonymous referees for their comments and suggestions, leading to many improvements in the paper.

References

[1] T. Aoto. Disproving Confluence of Term Rewriting Systems by Interpretation and Ordering. In *Proc. of FroCoS'13*, LNCS 8152:311-326, 2013.

[2] G.S. Boolos, J.P. Burgess, and R.C. Jeffrey. Computability and Logic, fourth edition. Cambridge University Press, 2002.

[3] R. Bruni and J. Meseguer. Semantic foundations for generalized rewrite theories. *Theoretical Computer Science* 351(1):386-414, 2006.

[4] J.-M. Cadiou. On Semantic Issues in the Relational Model of Data. In *Proc. of MFCS'76*, LNCS 45:23-38, 1976

[5] C.L. Chang and R.C. Lee. Symbolic Logic and Mechanical Theorem Proving. Academic Press, 1973.

[6] K.L. Clark. Negation as failure. In *Logic and Data Bases*, pages 293-322, 1978.

[7] K.L. Clark. Predicate Logic as a Computational Formalism. PhD. Thesis, Research Monograph 79/59 TOC, Department of Computing, Imperial College of Science, and Technology, University of London, December 1979.

[8] M. Clavel, F. Durán, S. Eker, P. Lincoln, N. Martí-Oliet, J. Meseguer, and C. Talcott. All About Maude – A High-Performance Logical Framework. LNCS 4350, Springer-Verlag, 2007.

[9] M. Clavel, M. Palomino, and A. Riesco. Introducing the ITP Tool: a Tutorial. *Journal of Universal Computer Science* 12(11):1618-1650, 2006.

[10] E.F. Codd. A Relational Model of Data for Large Shared Data Banks. *Communications of the ACM* 13(6):377-387, 1970.

[11] A. Colmerauer, H. Kanqui, P. Roussel, and R. Pasero. Un Systeme de Communication Homme-Machine en Francais. Rapport de recherche CRI 72-18, Groupe de rechereche en Intelligence Artificielle, U.E.R. de Luminy, Université d'Aix-Marseille, 1973.

[12] M. Dauchet and S. Tison. The Theory of Ground Rewrite Systems is Decidable. In *Proc. of LICS '90*, pages 242-248, IEEE Press, 1990

[13] N. Dershowitz, S. Kaplan, and D. Plaisted. Rewrite, rewrite, rewrite, rewrite, rewrite, . . . *Theoretical Computer Science* 83:71-96, 1991.

[14] M.H. van Emden and R.A. Kowalski. The semantics of Predicate Logic as a Programming Language. *Journal of the ACM* 23(4):733-742, 1976.

[15] J. Endrullis and D. Hendriks. Lazy productivity via termination. *Theoretical Computer Science* 412:3203-3225, 2011.

[16] R. Fagin. Functional Dependencies in a Relational Database and Propositional Logic. *IBM Journal of Research and Development* 21(6):543-544, 1977

[17] J.P. Gallagher and M. Rosendahl. Approximating Term Rewriting Systems: A Horn Clause Specification and Its Implementation. In *Proc. of LPAR 2008*, LNCS 5330:682-696, 2008.

[18] M. Gelfond. Answer Sets. In *Handbook of Knowledge Representation*, pages 285-316, Elsevier, 2008.

[19] M. Gelfond and V. Lifschitz. The Stable Model Semantics for Logic Programming. In *Proc. of ICLP/SLP'88*, volume 2, pages 1070-1080, The MIT Press, 1988.

[20] J. Goguen and J. Meseguer. Models and Equality for Logical Programming. In *Proc. of TAPSOFT'87*, LNCS 250:1-22, Springer-Verlag, 1987.

[21] J. Goubault-Larrecq. Finite Models for Formal Security Proofs. *Journal of Computer Security* 18(6):1247-1299, 2010.

[22] C.C. Green and B. Raphael. The use of theorem-proving techniques in question-answering systems. In *Proc. of the 1968 ACM National Conference*, pages 169-181, 1968.

[23] R. Gutiérrez and S. Lucas. Automatic Generation of Logical Models with AGES. In *Proc. of CADE 2019*, LNCS 11716:287-299, 2019. Tool page: `http://zenon.dsic.upv.es/ages/`.

[24] R. Gutiérrez and S. Lucas. infChecker, a tool for checking infeasibility. In *Proc. of IWC 2019*, pages 38-42, 2019. Tool page: `http://zenon.dsic.upv.es/infChecker/`

[25] W. Hodges. Model Theory. Cambridge University Press, 1993.

[26] J. Jürjens and T. Weber. Finite Models In FOL-Based Crypto-Protocol Verification. In *Revised Selected Papers from ARSPA-WITS 2009*, LNCS 5511:155-172, 2009.

[27] D. Kapur and D.R. Musser. Proof by Consistency. *Artificial Intelligence* 31:125-157, 1987.

[28] S. Kim and H. Zhang. ModGen: Theorem Proving by Model Generation. In *Proc. of AAAI'94*, pages 162-167, AAAI Press/MIT Press, 1994.

[29] A. Lisitsa. Finite Models vs. Tree Automata in Safety Verification. In *Proc. of RTA 2012*, LIPIcs 15:225–239, 2012.

[30] A. Lisitsa. Finite Reasons For Safety. Parameterized Verification By Finite Model Finding. *Journal of Automated Reasoning* 51:431-451, 2013.

[31] G. Lowe. Breaking and Fixing the Needham-Schroeder Public Key Protoco lUsing FDR. In *Proc. of TACAS'96*, LNCS 1055:147-166, 1996.

[32] S. Lucas. Context-Sensitive Rewriting Strategies. *Information and Computation* 178(1):293–343, 2002.

[33] S. Lucas. Analysis of Rewriting-Based Systems as First-Order Theories. In *Revised Selected papers from LOPSTR 2017*, LNCS 10855:180-197, 2018.

[34] S. Lucas. Proving Program Properties as First-Order Satisfiability. In *Revised Selected papers from LOPSTR 2018*, LNCS 11408:3-21, 2019.

[35] S. Lucas and R. Gutiérrez. Automatic Synthesis of Logical Models for Order-Sorted First-Order Theories. *Journal of Automated Reasoning* 60(4):465–501, 2018.

[36] S. Lucas and R. Gutiérrez. Use of logical models for proving infeasibility in term rewriting. *Information Processing Letters*, 136:90-95, 2018.

[37] S. Lucas, C. Marché, and J. Meseguer. Operational termination of conditional term rewriting systems. *Information Processing Letters* 95:446–453, 2005.

[38] Z. Manna. Properties of programs and the First-Order Predicate Calculus. *Journal of the ACM* 16(2):244-255, 1969.

[39] W. McCune Prover9 and Mace4. `http://www.cs.unm.edu/~mccune/prover9/`, 2005–2010.

[40] E. Mendelson. Introduction to Mathematical Logic. Fourth edition. Chapman & Hall, 1997.

[41] J. Meseguer. Membership algebra as a logical framework for equational specification. In *Proc. of WADT'97*, LNCS 1376:18–61, Springer-Verlag, 1998.

[42] J. Meseguer. Conditional rewriting logic as a unified model of concurrency. *Theoretical Computer Science* 96:73-155, 1992

[43] A. Middeldorp. Approximating Dependency Graphs Using Tree Automata Techniques. In *Proc. of IJCAR'01*, LNAI 2083:593–610, 2001.

[44] A. Middeldorp, J. Nagele, and K. Shintani. Confluence Competition 2019. In *Proc. of TACAS 2019, 25 Years of TACAS: TOOLympics*, LNCS 11429:25-40, 2019.

[45] R.C. Moore. *Logic and Representation.* Cambridge University Press, 1995.

[46] J.M. Nicolas and H. Gallaire. Data Base Theory vs. Interpretation. In *Logic and Data Bases*, pages 33-54, Plenum Press, New York, 1978

[47] E. Ohlebusch. *Advanced Topics in Term Rewriting.* Springer-Verlag, Apr. 2002.

[48] R. Ramakrishnan and J.D. Ullman. A survey of deductive database systems. *Journal of Logic Programming* 23(2):125-149, 1995.

[49] F. Rapp and A. Middeldorp. Automating the First-Order Theory of Rewriting for Left-Linear Right-Ground Rewrite Systems. In *Proc. of FSCD'16*, LIPIcs 52, Article No. 36; pp. 36:1-36:12, 2016.

[50] R. Reiter. On Closed World Data Bases. In Logic and Data Bases, pages 119–140, Plenum Press, 1978.

[51] R. Reiter. Equality and Domain Closure In First-Order Databases. *Journal of the ACM* 27(2):235-249, 1980.

[52] R. Reiter. Towards a Logical Reconstruction of Relational Database Theory. In *On conceptual moddeling: perspectives from artificial intelligence, databases, and programming languages*, pages 191-238, 1984.

[53] G. Reger, M. Suda, and A. Voronkov. Finding Finite Models in Multi-sorted First-Order Logic. In *Proc. of SAT 2016*, LNCS 9710:323-341, 2016.

[54] J.A. Robinson. A Machine-Oriented Logic Based on the ResolutionPrinciple. *Journal of the ACM* 12(1):23-41, 1965.

[55] M. Schmidt-Schauss. Computational Aspects Of An Order-Sorted Logic With Term Declarations. PhD Thesis, Fachbereich Informatik der Universität Kaiserslautern, April 1988.

[56] P. Selinger. Models for an Adversary-Centric Protocol Logic. In *Proc. of LACPV'01*, Electronic Notes in Theoretical Computer Science 55:69-87, 2001.

[57] Z. Somogyi, F. Henderson, and T. Conway. The execution algorithm of Mercury, an efficient purely declarative logic programming language. *Journal of Logic Programming* 29(1–3):17-64, 1996.

[58] T. Sternagel and A. Middeldorp. Infeasible Conditional Critical Pairs. In *Proc. of IWC'15*, pages 13–18, 2014.

[59] C. Sternagel and T. Sternagel. Certifying Confluence Of Almost Orthogonal CTRSs Via Exact Tree Automata Completion. In *Proc. of FSCD'16*, LIPIcs 52, Article No. 85; pp. 85:1–85:16, 2016.

[60] C. Sternagel and A. Yamada. Reachability Analysis for Termination and Confluence of Rewriting. In *Proc. of TACAS 2019*, part I, LNCS 11427:262-278, 2019.

[61] R. Thomason. Logic and Artificial Intelligence. *Stanford Encyclopedia of Philosophy*, 110 pages, 2018. `https://plato.stanford.edu/archives/win2018/entries/logic-ai/`

[62] R. Treinen. The first-order theory of linear one-step rewriting is undecidable. *Theoretical Computer Science* 208:179-190, 1998.

[63] H. Wang. Logic of many-sorted theories. *Journal of Symbolic Logic* 17(2):105-116, 1952.

[64] J. Zhang and H. Zhang. Generating Models by SEM (System Description). In *Proc. of CADE'96*, LNCS 1104:308-312, 1996.