Deciding the Inductive Validity of $\forall\exists^*$ Queries

Matthias Horbach
and Christoph Weidenbach

MPI–I–2009–RG1–001 May 2009

Authors' Addresses

Matthias Horbach and Christoph Weidenbach
Max Planck Institute for Informatics
Campus E1 4
66123 Saarbrücken
Germany

Publication Notes

This report is a preliminary version of an article intended for publication elsewhere.

Acknowledgements

Matthias Horbach and Christoph Weidenbach are supported by the German Transregional Collaborative Research Center SFB/TR 14 AVACS.

Abstract

We present a new saturation-based decidability result for inductive validity. Let Σ be a finite signature in which all function symbols are at most unary and let N be a satisfiable Horn clause set without equality in which all positive literals are linear. If $N \cup \{A_1, \ldots, A_n \rightarrow\}$ belongs to a finitely saturating clause class, then it is decidable whether a sentence of the form $\forall\exists^*(A_1 \wedge \ldots \wedge A_n)$ is valid in the minimal model of N.

Keywords

decidability, inductive reasoning, saturation-based theorem proving

Contents

1 Introduction

We consider the problem of deciding whether a formula ϕ holds in the minimal model of a given satisfiable Horn clause set N over a signature Σ, written $N \models_{Ind} \phi$, or whether $N \cup \{\neg\phi\}$ does not have a Herbrand model over Σ. If all signature symbols are at most unary and all positive literals in N are linear, even first-order unsatisfiability is still undecidable: Consider a Post correspondence problem over the alphabet $\{a, b\}$ with given word pairs (u_i, v_i). We model words by monadic terms over the unary function symbols a, b with empty word 0. Then the PCP has a solution iff the following Horn clause set is unsatisfiable:

$$\begin{aligned} &\rightarrow \mathrm{PCP}(0,0) \\ \mathrm{PCP}(x,y) &\rightarrow \mathrm{PCP}(u_i(x), v_i(y)) \\ \mathrm{PCP}(a(x), a(x)) &\rightarrow \\ \mathrm{PCP}(b(x), b(x)) &\rightarrow \end{aligned}$$

Equivalently, it has a solution iff

$$\begin{aligned} \{\rightarrow \mathrm{PCP}(0,0),\ \mathrm{PCP}(x,y) &\rightarrow \mathrm{PCP}(u_i(x), v_i(y))\} \\ &\models_{Ind} \exists x.\mathrm{PCP}(a(x), a(x)) \vee \mathrm{PCP}(b(x), b(x))\ . \end{aligned}$$

In this paper, we identify a range of classes of clause sets and of query formulas for which validity in the minimal model is decidable. The main result is as follows:

Let N be a satisfiable set of Horn clauses without equality over a finite signature Σ and let $\{A_1, \ldots, A_n\}$ a set of atoms over Σ, where

(1) all function symbols in Σ are at most unary,

(2) all positive literals in N are linear, i.e. every variable occurs at most once, and

(3) $N \cup \{A_1, \ldots, A_n \rightarrow\}$ belongs to a class that can be finitely saturated by ordered resolution (with variant subsumption and tautology deletion).

Then the problem

$$N \models_{Ind} \forall x.\exists y_1, \ldots, y_m.(A_1 \wedge \ldots \wedge A_n)$$

is decidable, where $x, y_1, \ldots, y_m$ are the variables in $A_1, \ldots, A_n$. There are no restrictions on purely negative clauses as well as no restrictions on the structure of the terms appearing in negative literals.

Instead of proving $N \models_{Ind} \forall x.\exists y_1, \ldots, y_m.(A_1 \wedge \ldots \wedge A_n)$, we refute $N \models_{Ind} \exists x.\forall y_1, \ldots, y_m.\neg(A_1 \wedge \ldots \wedge A_n)$. The result also holds for an arbitrary positive quantifier-free formula ϕ in disjunctive normal form $\bigvee_i \bigwedge_j A_{i,j}$ in place of $A_1 \wedge \ldots \wedge A_n$, i.e. the problem $N \models_{Ind} \forall\exists^* \phi$ is decidable as long as the clauses $A_{i,1}, \ldots, A_{i,n_i} \rightarrow$ can be finitely saturated together with N.

The first-order unsatisfiability problem for Horn classes satisfying conditions (1)–(2) is still undecidable, as the above encoding of the Post correspondence problem shows. Therefore, the basis of our decidability result is finite first-order saturation (3). The side conditions (1)–(2) as well as the restriction to variant subsumption and tautology deletion for the saturation process are needed for our current proof. The latter is not an essential restriction, since most decidability results based on saturation show termination by restricting the depth of the occurring terms and the number of variables in each clause, which corresponds exactly to a saturation modulo variants and tautologies (cf. e.g. [10]).

The proof of our result is constructive. We demonstrate it on the example clause set

$$\begin{aligned} N_G = \{ \quad & \rightarrow G(s(s(0)), s(0)) \ , \\ & G(x, y) \rightarrow G(s(x), s(y)) \quad , \\ & G(s(x), s(y)) \rightarrow G(x, y) \quad \} \end{aligned}$$

with query $\forall x.\exists y.G(y, x)$. In the minimal model of N_G, the relation G is the "one greater" relation on the naturals. The clause set N satisfies conditions (1)–(2) and can be finitely saturated by ordered resolution, generating one additional clause $\rightarrow G(s(0), 0)$. It can also be finitely saturated after adding $G(y, x) \rightarrow$.

In [12], we have presented a superposition-based calculus that is complete for unsatisfiability of queries of the form $\exists^*\forall^*(A_1, \ldots, A_n \rightarrow)$ with respect to minimal models of Horn clause sets. The basic idea of the calculus is to treat the existentially quantified variables via an additional constraint. The result is a constrained query clause of the form $\vec{v} \approx \vec{x} \,\|\, A_1, \ldots, A_n \rightarrow$ where the existential variables $\vec{v}$ are replaced in the constrained clause by fresh universally quantified variables $\vec{x}$. The generalized ordered resolution rule of the calculus takes care of the compatibility of constraints (Section 3). For our example, we obtain the constrained query clause $v \approx x \,\|\, G(y, x) \rightarrow$.

The saturation of N and the constrained query clause does not terminate in general, even with the additional conditions (1)–(3). For the above example, we can generate infinitely many non-redundant constrained clauses of the form $v{\approx}x\sigma^n \parallel G(y,x) \rightarrow$ and $v{\approx}x\sigma^n\tau \parallel \square$ for $\sigma^0 = \{x \mapsto x\}$, $\sigma^{n+1} = \{x \mapsto s(x\sigma^n)\}$ and $\tau = \{x \mapsto 0\}$. Therefore, we generalize our previously developed constraint language [12] to "regular" substitution expressions for the existentially quantified variables (Section 2). E.g. a constraint $v{\approx}x\sigma^*$ represents all possible constraints of the form $v{\approx}x\sigma^n$. Together with condition (3), this enables the termination of the query saturation process (Proposition 3.18). For this proof, we need the finite saturation result with variant subsumption and tautology deletion. For the above example, we obtain the two additional constrained clauses $v{\approx}x\sigma^* \parallel G(y,x) \rightarrow$ and $v{\approx}x\sigma^*\tau \parallel \square$ for $\sigma = \{x \mapsto s(x)\}$, $\tau = \{x \mapsto 0\}$.

What remains to be shown is that the substitutions in the constraints of all derived constrained empty clauses are covering, i.e. represent all possible instantiations for the variables $\vec{x}$: If this is the case, then the clause set does not have a Herbrand model over the given signature. The conjunction of all regular substitution expressions for the constrained empty clauses can be transformed into a monadic Horn clause set containing only universally reductive and linear clauses (Section 4.1). The initial substitution expressions are covering iff a certain predicate $P(\vec{x})$ introduced in the translation is the total relation in the minimal model of the generated Horn clause set. For our example, this translation results in the following Horn clauses:

$$\begin{aligned} &\rightarrow P_1(0) \\ P(x) &\rightarrow P_2(s(x)) \\ P_1(x) &\rightarrow P(x) \\ P_2(x) &\rightarrow P(x) \end{aligned}$$

Applying predicate completion [8], we can generate a Horn clause set for the complement of P, named $\check{P}$, for any Horn clause set generated from a substitution expression, such that P is total iff $\check{P}$ is empty in the minimal models of the respective Horn clause sets (Section 4.3). The clause set for $\check{P}$ does not contain function symbols in negative literals anymore. Moreover, because of the restriction of the signature to unary function symbols, the translation causes the clause set to contain monadic predicates only. These properties enable the final decidability of the emptiness of $\check{P}$ by ordered resolution (Theorem 4.17). The complement $\check{P}$ for our example is defined by

the clauses

$$\begin{aligned} &\rightarrow \check{P}_1(s(x)) \\ &\rightarrow \check{P}_2(0) \\ \check{P}(x) &\rightarrow \check{P}_2(s(x)) \\ \check{P}_1(x), \check{P}_2(x) &\rightarrow \check{P}(x) \end{aligned}$$

that belong to a class where emptiness is decidable by ordered resolution [16, 15]. For the above clause set, the theory of the relation $\check{P}$ is empty, hence $N \models_{Ind} \forall x.\exists y.G(y,x)$ holds. Note that such clauses can in general not be represented by tree automata (even with constraints).

To the best of our knowledge, our result is the first decidability result for inductive validity based on a finite saturation concept for Horn clauses. Related approaches to inductive reasoning based on superposition include the works of Ganzinger and Stuber [11] and Comon and Nieuwenhuis [8]. Both are also applicable to equational clauses, but did not lead to new decidability results. Other work in the area of automated inductive theorem proving includes the test set calculi of Bouhoula et al. [2, 3] and approaches via term rewrite systems by Caferra and Zabel [4] and Kapur [13, 9]. All these approaches consider only purely universal queries and do not admit quantifier alternations.

2 Preliminaries

We build on the notions of [1, 17] and shortly recall here the most important concepts as well as the specific extensions needed for the new calculus.

Terms and Clauses

Let $\Sigma = (\mathcal{P}, \mathcal{F})$ be a *signature* consisting of a set $\mathcal{P}$ of predicate symbols of fixed arity and a set $\mathcal{F}$ of function symbols of fixed arity, and let $X \cup V$ be an infinite set of variables such that X, V, and $\mathcal{F}$ are disjoint and V is finite. Elements of X are called *universal variables* and denoted as x, y, z, and elements of V are called *existential variables* and denoted as v.

We denote by $\mathcal{T}(\mathcal{F}, X)$ the set of all *terms* over $\mathcal{F}$ and X and by $\mathcal{T}(\mathcal{F})$ the set of all *ground terms* over $\mathcal{F}$. Throughout this article, we require that $\mathcal{T}(\mathcal{F})$ is infinite. To improve readability, term tuples $(t_1, \ldots, t_n)$ will often be denoted $\vec{t}$. The variables occurring in a term t or a term tuple $\vec{t}$ are denoted by $\mathrm{vars}(t)$ or $\mathrm{vars}(\vec{t})$, respectively.

An *atom* is an expression of the form $P(t_1, \ldots, t_n)$, where $P \in \mathcal{P}$ is a predicate symbol of arity n and $t_1, \ldots, t_n$ are terms. A *(standard universal) clause* is a pair of multisets of atoms, written $\Gamma \to \Delta$, interpreted as the conjunction of all atoms in the *antecedent* Γ implying the disjunction of all atoms in the *succedent* Δ. A clause is *Horn* if Δ contains at most one atom. In this case, Δ is called the *head* of the clause. The *empty clause* is denoted by $\square$. A Horn clause is *universally reductive* if all variables of the antecedent appear also in the succedent.

Substitution Expressions and Constrained Clauses

A *(basic) substitution* σ is a map from a finite set $X' \subseteq X$ of universal variables to $\mathcal{T}(\mathcal{F}, X)$.[1] The application of σ to a term t or a term tuple $\vec{t}$ is

[1] We stray here from the more common definition where a substitution σ is defined on all variables and its domain is the set of variables on which σ acts non-trivially. The

denoted by $t\sigma$ or $\vec{t}\sigma$, respectively. The substitution σ is *linear* if no variable occurs twice in the term set $\{x\sigma \mid x \in X'\}$. The *most general unifier* of two terms s, t is denoted by $\mathrm{mgu}(s, t)$.

Consider a signature containing the substitutions as constants and function symbols $\circ$ (*composition*), $|$ (*disjunction*), and * (*repetition*) of arity 2, 2 and 1, respectively. A term in over this signature is called a *substitution expression*. Substitution expressions are denoted as $\bar{\sigma}, \bar{\tau}$. The symbols $\circ$ and $|$ are written in infix notation, and * is written in postfix notation. We will often write $\bar{\sigma} \circ \bar{\tau}$ as $\bar{\sigma}\bar{\tau}$.

The *domain* $\mathrm{dom}(\bar{\sigma})$ and the *variable range* $\mathrm{VRan}(\bar{\sigma})$ of a substitution expression are defined inductively as follows:

$$\begin{aligned}
\mathrm{dom}(\sigma) &= \{x_1, \ldots, x_n\} \\
&\quad \text{where } \sigma : \{x_1, \ldots, x_n\} \to \mathcal{T}(\mathcal{F}, X) \\
\mathrm{dom}(\bar{\sigma} \circ \bar{\tau}) &= \mathrm{dom}(\bar{\sigma}) \\
\mathrm{dom}(\bar{\sigma}_1 | \bar{\sigma}_2) &= \mathrm{dom}(\bar{\sigma}_1) \cup \mathrm{dom}(\bar{\sigma}_2) \\
\mathrm{dom}(\bar{\sigma}^*) &= \mathrm{dom}(\bar{\sigma})
\end{aligned}$$

$$\begin{aligned}
\mathrm{VRan}(\sigma) &= \mathrm{vars}(x_1\sigma, \ldots, x_n\sigma) \\
&\quad \text{where } \sigma : \{x_1, \ldots, x_n\} \to \mathcal{T}(\mathcal{F}, X) \\
\mathrm{VRan}(\bar{\sigma} \circ \bar{\tau}) &= \mathrm{VRan}(\bar{\tau}) \\
\mathrm{VRan}(\bar{\sigma}_1 | \bar{\sigma}_2) &= \mathrm{VRan}(\bar{\sigma}_1) \cap \mathrm{VRan}(\bar{\sigma}_2) \\
\mathrm{VRan}(\bar{\sigma}^*) &= \mathrm{dom}(\bar{\sigma})
\end{aligned}$$

A *constrained clause* $v_1 \approx x_1\bar{\sigma}, \ldots, v_n \approx x_n\bar{\sigma} \parallel C$, also written $\vec{v} \approx \vec{x}\bar{\sigma} \parallel C$, consists of a sequence of equations $v_1 \approx x_1\bar{\sigma}, \ldots, v_n \approx x_n\bar{\sigma}$ called the *constraint* and a clause C, such that $\{v_1, \ldots, v_n\} = V$, $x_1, \ldots, x_n \in X$ are universal variables and $\bar{\sigma}$ is a substitution expression with domain $\{x_1, \ldots, x_n\}$.

We abbreviate $\vec{v} \approx \vec{x}\sigma \parallel C$ as $\vec{v} \approx \vec{x} \parallel C$ if σ is the identity substitution on $\{x_1, \ldots, x_n\}$.

Orderings

Any ordering $\prec$ on atoms can be extended to clauses in the following way. We consider clauses as multisets of occurrences of atoms. The occurrence of an atom A in the antecedent is identified with the multiset $\{A, A\}$; the occurrence of an atom A in the succedent is identified with the multiset $\{A\}$. Now we lift $\prec$ to atom occurrences as its multiset extension, and to clauses as the multiset extension of this ordering on atom occurrences.

---

reason is that we need a strong control over the domain when we relate substitutions and clauses in Section 4.

An occurrence of an atom A is *maximal* in a clause C if there is no occurrence of an atom in C that is strictly greater with respect to $\prec$ than the occurrence of A. It is *strictly maximal* in C if there is no other occurrence of an atom in C that is greater than or equal to the occurrence of A with respect to $\prec$.

Throughout this paper, we will assume a reduction ordering $\prec$ that is total on ground atoms.

Denotations and Models

The *denotation* of a substitution expression is defined as follows:

$$\begin{aligned}\llbracket\sigma\rrbracket &= \{\sigma\}\\ \llbracket\bar\sigma\bar\tau\rrbracket &= \{\sigma\tau \mid \sigma \in \llbracket\bar\sigma\rrbracket, \tau \in \llbracket\bar\tau\rrbracket\}\\ \llbracket\bar\sigma_1|\bar\sigma_2\rrbracket &= \llbracket\bar\sigma_1\rrbracket \cup \llbracket\bar\sigma_2\rrbracket\\ \llbracket\bar\sigma^*\rrbracket &= \textstyle\bigcup_{n\geq 0}\llbracket\bar\sigma^n\rrbracket\end{aligned}$$

Here $\bar\sigma^0$ denotes the substitution $\{x \mapsto x \mid x \in \operatorname{dom}\bar\sigma\}$, and $\bar\sigma^{n+1} = \bar\sigma \circ \bar\sigma^n$. For those substitution expressions we will be interested in (cf. Section 4.1), $\operatorname{VRan}(\bar\sigma)$ is the set of all variables that appear in the image of each element of $\llbracket\bar\sigma\rrbracket$.

Because of the associativity of substitution composition and of set union, $\llbracket(\bar\sigma_1\bar\sigma_2)\bar\sigma_3\rrbracket = \llbracket\bar\sigma_1(\bar\sigma_2\bar\sigma_3)\rrbracket$ and $\llbracket(\bar\sigma_1|\bar\sigma_2)|\bar\sigma_3\rrbracket = \llbracket\bar\sigma_1|(\bar\sigma_2|\bar\sigma_3)\rrbracket$, i.e. $\circ$ and $|$ are associative. Hence we will identify $(\bar\sigma_1\bar\sigma_2)\bar\sigma_3$ and $\bar\sigma_1(\bar\sigma_2\bar\sigma_3)$, writing both as $\bar\sigma_1\bar\sigma_2\bar\sigma_3$ (and analogously for $|$).

Moreover, we define $\vec t\llbracket\bar\sigma\rrbracket = \{\vec t\sigma \mid \sigma \in \llbracket\bar\sigma\rrbracket\}$. A substitution expression $\bar\sigma$ with domain $\{x_1, \ldots, x_n\}$ is *covering for a set* $T \subseteq \mathcal{T}(\mathcal{F}, X)^n$ if all ground instances of elements of T are instances of an element of $(x_1, \ldots, x_n)\llbracket\bar\sigma\rrbracket$. If $\bar\sigma$ is covering for $\mathcal{T}(\mathcal{F})^n$, we say that $\bar\sigma$ is *covering*.

Two constrained clauses $\vec v{\approx}\vec x\bar\sigma \parallel C$ and $\vec v{\approx}\vec x\bar\sigma' \parallel C'$ are *variants* if there is a variable renaming $\pi : \operatorname{VRan}(\bar\sigma) \cup \operatorname{vars}(C) \to \operatorname{VRan}(\bar\sigma') \cup \operatorname{vars}(C')$ such that π maps the variables of $\operatorname{VRan}(\bar\sigma)$ to $\operatorname{VRan}(\bar\sigma')$, $C\pi = C'$, and $\llbracket\bar\sigma\pi\rrbracket = \llbracket\bar\sigma'\rrbracket$. If both C and C' are unconstrained, this reduces to the usual notion of variants. Note that the denotations of variants agree up to renaming of bound variables. If $\bar\sigma$ is a variable renaming and C does not contain any variables of $\vec v{\approx}\vec x\bar\sigma$, then we abbreviate the constrained clause as $\parallel C$. We call a constrained clause $\parallel C$ *unconstrained* and identify it with its clausal part C.

For constrained clause $\vec v{\approx}\vec x\bar\sigma \parallel C$, let $\llbracket\vec v{\approx}\vec x\bar\sigma \parallel C\rrbracket$ be the (potentially infinite) formula set $\llbracket\vec v{\approx}\vec x\bar\sigma \parallel C\rrbracket = \{\forall\vec y.\vec v{\approx}\vec x\sigma \to C \mid \sigma \in \llbracket\bar\sigma\rrbracket\}$, where the universal quantifier ranges over the variables of $\vec x\sigma$ and C. For a set N of constrained clauses, let $\llbracket N\rrbracket = \bigcup_{\vec v{\approx}\vec x\bar\sigma \parallel C \in N}\llbracket\vec v{\approx}\vec x\bar\sigma \parallel C\rrbracket$.

An interpretation $\mathcal{I}$ is said to *model* N, written $\mathcal{I} \models N$, if and only if the formula $\exists \vec{v}. \bigwedge_{\phi \in [\![N]\!]} \phi$ is valid in $\mathcal{I}$. In this case, $\mathcal{I}$ is called a *model* of N. A constrained clause set is *satisfiable* if it has a model.

If M and N are two constrained clause sets, we write $N \models M$ if each model of N is also a model of M. If N is satisfiable and Horn, we write $N \models_{Ind} M$ if the minimal model of N models M.

Inferences and Redundancy

An *inference rule* is a relation on constrained clauses. Its elements are called *inferences* and written as

$$\frac{\alpha_1 \parallel C_1 \quad \dots \quad \alpha_k \parallel C_k}{\alpha \parallel C} \quad .$$

The constrained clauses $\alpha_1 \parallel C_1, \dots, \alpha_k \parallel C_k$ are called the *premises* and $\alpha \parallel C$ the *conclusion* of the inference. An *inference calculus* is a set of inference rules.

A constrained clause $\vec{v} \approx \vec{x}\bar{\sigma} \parallel C$ is *redundant* with respect to a constrained clause set N if C is a tautology or if there is a variant $\vec{v} \approx \vec{x}\bar{\tau} \parallel C$ of a constrained clause in N such that $[\![\bar{\sigma}]\!] \subsetneq [\![\bar{\tau}]\!]$.[2] An inference is called *redundant* with respect to N if its conclusion is redundant with respect to N or if a premise C is redundant with respect to $N \setminus \{C\}$. A constrained clause set N is *saturated* with respect to a given inference calculus if each inference in the calculus with premises in N is redundant with respect to N.

A *derivation* is a finite or infinite sequence $N_0, N_1, \dots$ such that for each i, there is an inference with premises in N_i and conclusion $\vec{v} \approx \vec{x}\bar{\sigma} \parallel C$ that is not redundant wrt. N_i, such that $N_{i+1} = N_i \cup \{\vec{v} \approx \vec{x}\bar{\sigma} \parallel C\}$.

[2] Usually, redundancy is defined based on a well-founded clause ordering. The notion of redundancy used in this article is not well-founded, as, e.g., each constrained clause in the sequence $\vec{v} \approx \vec{x}\sigma_1 \parallel C$, $\vec{v} \approx \vec{x}(\sigma_1|\sigma_2) \parallel C$, $\vec{v} \approx \vec{x}(\sigma_1|\sigma_2|\sigma_3) \parallel C$, ... is redundant with respect to its successor. This is why we need a different argument to ensure the termination of saturation (cf. Section 3.4).

3 A Calculus for Constrained Clauses

We will now introduce an inference system for constrained clauses. This inference system is based on a calculus presented in [12], which allows also constrained clauses containing equational atoms. We will show how this superposition-based calculus, which we will call SFD, behaves in our setting and how to enrich it by a constraint melting inference rule, which makes the calculus terminating for a class of constrained Horn clauses.

We present the calculus SFD in Section 3.1. It can be used to examine queries of the form $N \models_{Ind} \forall\vec{x}.\exists\vec{y}.\phi$, where ϕ is a quantifier-free formula and N is a set of clauses, and both N and the query may contain equations. The basic idea is to use a superposition-style approach to examine whether $N \cup \{\exists\vec{x}.\forall\vec{y}.\neg\phi\} \models_{Ind} \square$. For this, N and the negated query are expressed as a set of constrained clauses, such that the constraint part allows a special treatment of the existential variables without Skolemization.

Example 3.1. In the example of the "one greater" relation presented in the introduction, where the set of existential variables is $V = \{v\}$, the correspondence between the components of the initial problem and their respective representations as constrained clauses is as follows:

$$\begin{array}{rcl}
\rightarrow G(s(s(0)), s(0)) & \hat{=} & v{\approx}x \parallel \rightarrow G(s(s(0)), s(0)) \\
G(x,y) \rightarrow G(s(x), s(y)) & \hat{=} & v{\approx}x \parallel G(y_1,y_2) \rightarrow G(s(y_1), s(y_2)) \\
G(s(x), s(y)) \rightarrow G(x,y) & \hat{=} & v{\approx}x \parallel G(s(y_1), s(y_2)) \rightarrow G(y_1,y_2) \\
\text{negation of } \forall x.\exists y.G(y,x) & \hat{=} & v{\approx}x \parallel G(y,x) \rightarrow
\end{array}$$

Since we omit constraints whose variables do not occur in the clausal part (cf. the definition of constrained clauses), we can also write this as follows:

$$\begin{array}{rcl}
\rightarrow G(s(s(0)), s(0)) & \hat{=} & \parallel \rightarrow G(s(s(0)), s(0)) \\
G(x,y) \rightarrow G(s(x), s(y)) & \hat{=} & \parallel G(y_1,y_2) \rightarrow G(s(y_1), s(y_2)) \\
G(s(x), s(y)) \rightarrow G(x,y) & \hat{=} & \parallel G(s(y_1), s(y_2)) \rightarrow G(y_1,y_2) \\
\text{negation of } \forall x.\exists y.G(y,x) & \hat{=} & v{\approx}x \parallel G(y,x) \rightarrow
\end{array}$$

As we will see, SFD does not necessarily terminate on $N \cup \{\exists\vec{x}.\forall\vec{y}.\neg\phi\}$. In Section 3.2, we will identify an adaption of SFD using a constraint melting rule, called QRM, that allows to decide the class of so-called existential query problems. Soundness, completeness and termination of QRM for existential query problems are proved in Sections 3.3 and 3.4.

3.1 The Fixed Domain Calculus SFD

In [12], we introduced a calculus for reasoning in fixed domains. I.e., instead of preserving first-order (un-)satisfiability, this calculus operates on a given constrained clause set in such a way that the (non-)existence of a Herbrand model over a given signature is preserved and can easily be checked for a saturated set.

This *superposition calculus for fixed domains (SFD)* allows equational atoms, but it only allows basic substitutions in constraints. Hence a constraint $\vec{v}\approx\vec{x}\sigma$ can be written as $\vec{v}\approx\vec{t}$, where $t_i = x_i\sigma$. Predicates are encoded as equations as usual, i.e. $P(\vec{t})$ is encoded as $f_P(\vec{t})\approx\text{true}$. So for the rest of this section only, all appearing constrained clauses will be supposed to contain only basic substitutions in constraints and only equational atoms.

If $\alpha_1 = \vec{v}\approx\vec{s}$ and $\alpha_2 = \vec{v}\approx\vec{t}$ are two constraints, then we write $\alpha_1\approx\alpha_2$ for the equations $s_1\approx t_1, \ldots, s_n\approx t_n$, which do not contain any existential variables, and we write $\text{mgu}(\alpha_1, \alpha_2)$ for the most general simultaneous unifier of the pairs $(s_1, t_1), \ldots, (s_n, t_n)$.

The calculus SFD consists of the following inference rules:

Equality Resolution:

$$\frac{\alpha \,\|\, \Gamma, s\approx t \to \Delta}{(\alpha \,\|\, \Gamma \to \Delta)\sigma}$$

where (1) $\sigma = \text{mgu}(s, t)$ and (2) $(s\approx t)\sigma$ is maximal in $(\Gamma, s\approx t \to \Delta)\sigma$.

Equality Factoring:

$$\frac{\alpha \,\|\, \Gamma \to \Delta, s\approx t, s'\approx t'}{(\alpha \,\|\, \Gamma, t\approx t' \to \Delta, s'\approx t')\sigma}$$

where (1) $\sigma = \text{mgu}(s, s')$, (2) $(s\approx t)\sigma$ is maximal (as an atom occurrence) in $(\Gamma \to \Delta, s\approx t, s'\approx t')\sigma$, and (3) $t\sigma \not\succeq s\sigma$.

Superposition, Right:

$$\frac{\alpha_1 \,\|\, \Gamma_1 \to \Delta_1, l\approx r \quad \alpha_2 \,\|\, \Gamma_2 \to \Delta_2, s[l']_p\approx t}{(\alpha_1 \,\|\, \Gamma_1, \Gamma_2 \to \Delta_1, \Delta_2, s[r]_p\approx t)\sigma_1\sigma_2}$$

where (1) $\sigma_1 = \mathrm{mgu}(l, l')$, $\sigma_2 = \mathrm{mgu}(\alpha_1\sigma_1, \alpha_2\sigma_1)$ and $\mathrm{dom}(\sigma_2) \cap V = \emptyset$, (2) $(l{\approx}r)\sigma_1\sigma_2$ is strictly maximal in $(\Gamma_1 \to \Delta_1, l{\approx}r)\sigma_1\sigma_2$ and $(s{\approx}t)\sigma_1\sigma_2$ is strictly maximal in $(\Gamma_2 \to \Delta_2, s{\approx}t)\sigma_1\sigma_2$, (3) $r\sigma_1\sigma_2 \not\succeq l\sigma_1\sigma_2$ and $t\sigma_1\sigma_2 \not\succeq s\sigma_1\sigma_2$, and (4) l' is not a variable.

Superposition, Left:

$$\frac{\alpha_1 \parallel \Gamma_1 \to \Delta_1, l{\approx}r \quad \alpha_2 \parallel \Gamma_2, s[l']_p{\approx}t \to \Delta_2}{(\alpha_1 \parallel \Gamma_1, \Gamma_2, s[r]_p{\approx}t \to \Delta_1, \Delta_2)\sigma_1\sigma_2}$$

where (1) $\sigma_1 = \mathrm{mgu}(l, l')$, $\sigma_2 = \mathrm{mgu}(\alpha_1\sigma_1, \alpha_2\sigma_1)$ and $\mathrm{dom}(\sigma_2) \cap V = \emptyset$, (2) $(l{\approx}r)\sigma_1\sigma_2$ is strictly maximal in $(\Gamma_1 \to \Delta_1, l{\approx}r)\sigma_1\sigma_2$, $(s{\approx}t)\sigma_1\sigma_2$ is maximal in $(\Gamma_2 \to \Delta_2, s{\approx}t)\sigma_1\sigma_2$, (3) $r\sigma_1\sigma_2 \not\succeq l\sigma_1\sigma_2$ and $t\sigma_1\sigma_2 \not\succeq s\sigma_1\sigma_2$, and (4) l' is not a variable.

Constraint Superposition:

$$\frac{\alpha_1 \parallel \Gamma_1 \to \Delta_1, l{\approx}r \quad v{\approx}t[l']_p, \alpha_2 \parallel \Gamma_2 \to \Delta_2}{(v{\approx}t[r]_p, \alpha_2 \parallel \alpha_1{\approx}(v{\approx}t[r]_p, \alpha_2), \Gamma_1, \Gamma_2 \to \Delta_1, \Delta_2)\sigma}$$

where (1) $\sigma = \mathrm{mgu}(l, l')$, (2) $(l{\approx}r)\sigma$ is strictly maximal in $(\Gamma_1 \to \Delta_1, l{\approx}r)\sigma$, (3) $r\sigma \not\succeq l\sigma$, and (4) l' is not a variable.

Equality Elimination:

$$\frac{\alpha_1 \parallel \Gamma \to \Delta, l{\approx}r \quad v{\approx}t[r']_p, \alpha_2 \parallel \square}{(\alpha_1 \parallel \Gamma \to \Delta)\sigma_1\sigma_2}$$

where (1) $\sigma_1 = \mathrm{mgu}(r, r')$, $\sigma_2 = \mathrm{mgu}(\alpha_1\sigma_1, (v{\approx}t[l]_p, \alpha_2)\sigma_1)$ and $V \cap \mathrm{dom}(\sigma_2) = \emptyset$, (2) $(l{\approx}r)\sigma_1\sigma_2$ is strictly maximal in $(\Gamma \to \Delta, l{\approx}r)\sigma_1\sigma_2$, (3) $r\sigma_1\sigma_2 \not\succeq l\sigma_1\sigma_2$, and (4) r' is not a variable.

This calculus is both sound and complete for fixed domain reasoning, which, in the case of constrained Horn clauses, coincides with inductive reasoning:

Proposition 3.2 (Soundness [12, Proposition 3])**.** *Let $\alpha \parallel C$ be the conclusion of an inference with premises in a constrained clause set N. Then $N \models_{Ind} N \cup \{\alpha \parallel C\}$.*

Theorem 3.3 (Completeness [12, Theorem 1]). *Let N be a finite set of constrained Horn clauses that is saturated*[1] *with respect to the calculus SFD and let $\vec{v}{\approx}\vec{x}\sigma_1 \parallel \Box, \ldots, \vec{v}{\approx}\vec{x}\sigma_m \parallel \Box$ be the constrained clauses in N with empty clausal part. Then N has a Herbrand model iff $\sigma_1 | \ldots | \sigma_m$ is not covering.*[2]

3.2 Melting and the Calculus QRM

Now let us go back to our non-equational Horn setting. The negation of a query of the form $\forall\vec{x}.\exists\vec{y}.\phi$ with purely positive and quantifier-free ϕ corresponds to a set of constrained clause of the form $\vec{v}{\approx}\vec{x} \parallel \Gamma \rightarrow$, i.e. with empty succedent. This motivates the following definition:

Definition 3.4. An *existential query problem* is a the set of constrained Horn clauses all of which are of the form

- $\parallel \Gamma \rightarrow A$ or
- $\vec{v}{\approx}\vec{x} \parallel \Gamma \rightarrow$.

The only useful rule of SFD in a non-equational setting is left superposition (followed by an elimination of the equation true≈true in the antecedent): Equality resolution, constraint superposition, and equality elimination are never applicable if all atoms are of the form $f_P(\vec{t}){\approx}$true. Equality factoring and right superposition are applicable, but they only create clauses that contain a positive atom true≈true and are tautologies.

When we do inferences between clauses in an equational query problem, the resulting clauses are still either unconstrained or have an empty succedent, i.e. are of the form

- $\parallel \Gamma \rightarrow A$ or
- $\vec{v}{\approx}\vec{x}\sigma \parallel \Gamma \rightarrow$.

[1] We actually considered a more general redundancy criterion: A ground constrained clause $\alpha \parallel C$ (with only a basic substitution in the constraint), i.e. one that does not contain universal variables, is called *redundant* with respect to a set N of constrained clauses if there are ground instances $\alpha \parallel C_1, \ldots, \alpha \parallel C_k$ of constrained clauses in N such that $C_i \prec C$ for all i and $C_1, \ldots, C_k \models C$. A non-ground constrained clause is redundant if all its ground instances are redundant.

If we identify a general constrained clause $\vec{v}{\approx}\vec{x}\bar{\sigma} \parallel C$ with the set $\{(\vec{v}{\approx}\vec{x}\sigma \parallel C) \mid \sigma \in [\![\bar{\sigma}]\!]\}$, our current redundancy notion is a restriction thereof.

[2] In fact, the completeness theorem [12, Theorem 1] also includes the case of infinite clause sets and an infinite set $\vec{v}{\approx}\vec{x}\sigma_1 \parallel \Box, \vec{v}{\approx}\vec{x}\sigma_2 \parallel \Box, \ldots$ of constrained empty clauses. However, the corresponding coverage property corresponds to an infinite disjunction $\sigma_1 | \sigma_2 | \ldots$ and hence cannot be expressed with our current terminology.

Hence the first premise in a left superposition inference always has an empty constraint, while the second premise may be constrained.

We write down this rule in its predicative form and adapt it to this setting and to general constraints:

Ordered Query Resolution:

$$\frac{\Gamma_1 \to A_1 \quad \vec{v}{\approx}\vec{x}\bar{\sigma} \parallel \Gamma_2, A_2 \to \Delta_2}{\vec{v}{\approx}\vec{x}\bar{\sigma}\tau' \parallel \Gamma_1\tau, \Gamma_2\tau \to \Delta_2\tau}$$

where (1) τ is the most general unifier of A_1 and A_2, (2) $\tau' : \mathrm{VRan}(\bar{\sigma}) \to \mathcal{T}(\mathcal{F}, X)$ maps y to $y\tau$ if $y \in \mathrm{dom}(\tau)$ and to y otherwise, and (3) $A_1\tau$ is strictly maximal in $(\Gamma_1 \to A_1)\tau$ and $A_2\tau$ is maximal in $(\Gamma_2, A_2 \to \Delta_2)\tau$, where Δ_2 is either empty or contains a single atom.

Note that ordered query resolution can as usual be restricted by means of a literal selection function. Moreover, the usual (unconstrained) ordered resolution rule is included as the restriction of ordered query resolution to unconstrained clauses.

The main weakness of the calculus SFD, and also of ordered query resolution alone, is that it does not terminate even in very simple cases.

Example 3.5. Consider again our example of the "one greater" relation defined by the unconstrained clauses

$$\begin{aligned} &\to G(s(s(0)), s(0)) \\ G(x, y) &\to G(s(x), s(y)) \\ G(s(x), s(y)) &\to G(x, y) \end{aligned}$$

and the query $\forall x.\exists y.G(y, x)$.

If we start from the translation of the problem into the language of constrained clauses in Example 3.1, a possible derivation with ordered query resolution (or with SFD) runs as follows. Remember that we omit constraints

whose variables do not occur in the clausal part.

$$
\begin{array}{rlll}
\text{clauses defining } G\text{: } 1: & & \| & \rightarrow G(s(s(0)), s(0)) \\
2: & & \| & G(y_1, y_2) \rightarrow G(s(y_1), s(y_2)) \\
3: & & \| & G(s(y_1), s(y_2)) \rightarrow G(y_1, y_2) \\
\text{negated query: } 4: & v{\approx}x & \| & G(y, x) \rightarrow \\
\text{resolve}(1{,}3) = 5: & & \| & \rightarrow G(s(0), 0) \\
\text{resolve}(5{,}4) = 6: & v{\approx}0 & \| & \square \\
\text{resolve}(2{,}4) = 7: & v{\approx}s(x) & \| & G(y, x) \rightarrow \\
\text{resolve}(5{,}7) = 8: & v{\approx}s(0) & \| & \square \\
\text{resolve}(2{,}7) = 9: & v{\approx}s(s(x)) & \| & G(y, x) \rightarrow \\
\text{resolve}(5{,}9) = 10: & v{\approx}s(s(0)) & \| & \square \\
\text{resolve}(2{,}9) = 11: & v{\approx}s(s(x)) & \| & G(y, x) \rightarrow \\
\text{resolve}(5{,}11) = 12: & v{\approx}s(s(s(0))) & \| & \square \\
& \ldots & &
\end{array}
$$

Clause 2 can now be resolved with clause 11 (and its descendants) an arbitrary number of times, and clause 5 can be resolved with each of them. So similar constrained clauses will now be computed with ever increasing constraints, making the derivation non-terminating.

A human observer of the previous example will quickly see that constrained clauses with empty clausal part, i.e. contradictions, will be derived successively for all constraints of the form $v{\approx}s^n(0)$, where $s^n(0)$ denotes the n-fold application $s(\ldots s(0)\ldots)$ of s to 0. He might thus replace all these infinitely many constrained clauses by only one constrained "super" clause, writing it, e.g., as $v{\approx}s^*(0) \,\|\, \square$. In fact, we will see that such repetitive behavior is the only reason for non-termination, and that the melting of the repeated constrained clauses into one constrained clause subsuming them all can be automated.

To do so, we now extend the calculus SFD to an inference system consisting of ordered query resolution and a *melting* rule. To define melting, we need the notion of an *ancestor*. In an ordered query resolution inference, the ancestors of the conclusion are the rightmost premise and all of its ancestors.

Melting:

$$\frac{\vec{v}{\approx}\vec{x}\bar{\sigma} \,\|\, C \quad \vec{v}{\approx}\vec{x}\bar{\sigma}\bar{\tau}' \,\|\, C'}{\vec{v}{\approx}\vec{x}\bar{\sigma}'' \,\|\, C}$$

if (1) $\vec{v}{\approx}\vec{x}\bar{\sigma} \,\|\, C$ is an ancestor of $\vec{v}{\approx}\vec{x}\bar{\sigma}\bar{\tau}' \,\|\, C'$, (2) there is a substitution expression $\bar{\tau}$ such that $\vec{v}{\approx}\vec{x}\bar{\sigma}\bar{\tau}' \,\|\, C'$ is a variant of $\vec{v}{\approx}\vec{x}\bar{\sigma}\bar{\tau} \,\|\, C$, and either (3.i) $\bar{\sigma}$ is of the form $\bar{\sigma} = \bar{\sigma}_1\bar{\sigma}_2^*$ and $\bar{\sigma}'' = \bar{\sigma}_1(\bar{\sigma}_2|\bar{\tau})^*$, or (3.ii) $\bar{\sigma}$ is not of this form and $\bar{\sigma}'' = \bar{\sigma}\bar{\tau}^*$.

A constrained clause that can serve as the rightmost premise of a melting inference is called *meltable*. The leftmost premise of a melting inference is called the *base clause* for the melting, and the conclusion the *melted clause*. The ancestors of the conclusion are defined as the ancestors of the base clause.

Definition 3.6. The calculus consisting of the rules "Ordered Query Resolution" and "Melting" is called *QRM*.

Example 3.7. A possible QRM derivation for our introductory example looks as follows, where $\sigma = \{x \mapsto s(x)\}$ and $\tau = \{x \mapsto 0\}$:

$$\begin{array}{rlll}
\text{clauses defining } G\text{: } 1: & & \| & \to G(s(s(0)), s(0)) \\
2: & & \| & G(x,y) \to G(s(x), s(y)) \\
3: & & \| & G(s(x), s(y)) \to G(x,y) \\
\text{negated query: } 4: & v{\approx}x & \| & G(y,x) \to \\
\text{resolve}(1,3) = 5: & & \| & \to G(s(0), 0) \\
\text{resolve}(5,4) = 6: & v{\approx}x\tau & \| & \square \\
\text{resolve}(2,4) = 7: & v{\approx}x\sigma & \| & G(y,x) \to \\
\text{melt}(4,7) = 8: & v{\approx}x\sigma^* & \| & G(y,x) \to \\
\text{resolve}(5,8) = 9: & v{\approx}x\sigma^*\tau & \| & \square
\end{array}$$

The only constrained clauses with non-empty ancestor set are the constrained clauses 5, 6, 7, and 9, with ancestors {3}, {4}, {4}, and {8}, respectively. The constrained clause set {1,...,9} is saturated with respect to QRM.

3.3 Soundness and Completeness

To establish the soundness of QRM, we have to show the soundness of the two rules ordered query resolution and melting. While the former is easy, the main objective will be to prove that all elements of the conclusion's denotation in a melting inference step are really consequences of the premises.

Lemma 3.8 (Soundness of Ordered Query Resolution)**.** *The ordered query resolution rule is sound.*

Proof. For an inference

$$\frac{\Gamma_1 \to A_1 \quad \vec{v}{\approx}\vec{x}\bar{\sigma} \,\|\, \Gamma_2, A_2 \to \Delta_2}{\vec{v}{\approx}\vec{x}\bar{\sigma}\tau' \,\|\, \Gamma_1\tau, \Gamma_2\tau \to \Delta_2\tau}$$

to be sound, it suffices that each inference

$$\frac{\Gamma_1 \to A_1 \quad \vec{v}{\approx}\vec{x}\sigma \,\|\, \Gamma_2, A_2 \to \Delta_2}{\vec{v}{\approx}\vec{x}\sigma\tau' \,\|\, \Gamma_1\tau, \Gamma_2\tau \to \Delta_2\tau}$$

for $\sigma \in [\![\bar{\sigma}]\!]$ is sound. This is the case because of the soundness of the SFD rules from [12], c.f. Proposition 3.2. □

The soundness of melting is not that obvious: The melting rule introduces constrained clauses that do not directly follow from the premises. For example, the constrained clauses $v{\approx}x \parallel G(y,x) \to$ and $v{\approx}x\sigma \parallel G(y,x) \to$ by themselves do not imply $v{\approx}x\sigma^* \parallel G(y,x) \to$.

In fact, the soundness of melting strongly depends on the context, e.g. on other available inference rules and the clauses it is applied to. Above all, the requirement that the left-most premise of ordered query resolution inferences is unconstrained, is imperative.

Example 3.9. Assume that we allowed ordered query resolution inferences in the style of SFD, i.e. with a constrained first premise. Consider an ordered query resolution inference between the two clauses $v{\approx}s(x) \parallel P(x) \to P(s(x))$ and $v{\approx}x \parallel P(x) \to$:

$$\frac{v{\approx}s(x) \parallel P(x) \to P(s(x)) \quad v{\approx}x \parallel P(x) \to}{v{\approx}s(x) \parallel P(x) \to}$$

The application of melting to $v{\approx}x \parallel P(x) \to$ and $v{\approx}s(x) \parallel P(x) \to$ is unsound, because $v{\approx}s(s(x)) \parallel P(x) \to$ is not implied by the given constrained clauses.

To establish the soundness of melting for existential query problems, we make two observations:

Since the leftmost premise of each ordered query resolution inference is unconstrained (by definition), a query resolution inference with rightmost premise $\vec{v}{\approx}\vec{x}\bar{\sigma} \parallel C$ and conclusion $\vec{v}{\approx}\vec{x}\bar{\sigma}\tau \parallel D$ can also be made with any other constrained clause $\vec{v}{\approx}\vec{x}\bar{\sigma}' \parallel C$ with the same clausal part but a different constraint, then resulting in the constrained clause $\vec{v}{\approx}\vec{x}\bar{\sigma}'\tau \parallel D$:

$$\frac{\Gamma \to A \quad v{\approx}x\bar{\sigma} \parallel C}{\vec{v}{\approx}\vec{x}\bar{\sigma}\tau \parallel D} \qquad \rightsquigarrow \qquad \frac{\Gamma \to A \quad v{\approx}x\bar{\sigma}' \parallel C}{\vec{v}{\approx}\vec{x}\bar{\sigma}'\tau \parallel D}$$

If the former inference is sound, so is the latter.

The second observation is that, if a derivation starts from an existential query problem, i.e. from constrained clauses that do not contain any stars, then all stars appearing in constraints during the derivation come from melting steps.

Lemma 3.10 (Soundness of Melting)**.** *For derivations starting from an existential query problem, the melting rule is sound.*

Proof. Consider a derivation step from a constrained clause set N to N' where a melting inference

$$\frac{\vec{v}{\approx}\vec{x}\bar{\sigma}_1\bar{\sigma}_2^* \parallel C \quad \vec{v}{\approx}\vec{x}\bar{\sigma}_1\bar{\sigma}_2^*\bar{\tau}' \parallel C'}{\vec{v}{\approx}\vec{x}\bar{\sigma}_1(\bar{\sigma}_2|\bar{\tau})^* \parallel C}$$

is performed. We will show that, for each integer $n \geq 0$, $\vec{v}{\approx}\vec{x}\bar{\sigma}_1(\bar{\sigma}_2|\bar{\tau})^n \parallel C$ is implied by the constrained clauses in N.

Since the derivation started from constrained clauses whose constraints do not contain any stars, the constrained clause $\vec{v}{\approx}\vec{x}\bar{\sigma}_1\bar{\sigma}_2^* \parallel C$ must have been derived from a constrained clause $\vec{v}{\approx}\vec{x}\bar{\sigma}_1 \parallel C$ to account for the star around $\bar{\sigma}_2$.

So the case $n = 0$ is trivial.

If $n > 0$, assume that the constrained clause $\vec{v}{\approx}\vec{x}\bar{\sigma}_1(\bar{\sigma}_2|\bar{\tau})^{n-1} \parallel C$ is implied. Moreover, we may inductively assume that all previous steps in the derivation are sound.

Following the remarks preceding this lemma, we could do the same set of inference steps needed to derive $\vec{v}{\approx}\vec{x}\bar{\sigma}_1\bar{\sigma}_2^* \parallel C$ from $\vec{v}{\approx}\vec{x}\bar{\sigma}_1 \parallel C$ to derive $\vec{v}{\approx}\vec{x}\bar{\sigma}_1(\bar{\sigma}_2|\bar{\tau})^{n-1}\bar{\sigma}_2^* \parallel C$ from $\vec{v}{\approx}\vec{x}\bar{\sigma}_1(\bar{\sigma}_2|\bar{\tau})^{n-1} \parallel C$. This constrained clause directly implies $\vec{v}{\approx}\vec{x}\bar{\sigma}_1(\bar{\sigma}_2|\bar{\tau})^{n-1}\bar{\sigma}_2 \parallel C$.

Moreover, we could do the same set of inference steps needed to derive $\vec{v}{\approx}\vec{x}\bar{\sigma}_1\bar{\sigma}_2^*\bar{\tau} \parallel C$ from $\vec{v}{\approx}\vec{x}\bar{\sigma}_1\bar{\sigma}_2^* \parallel C$ to derive $\vec{v}{\approx}\vec{x}\bar{\sigma}_1(\bar{\sigma}_2|\bar{\tau})^{n-1}\bar{\tau} \parallel C$.

Thus, we know that both $\vec{v}{\approx}\vec{x}\bar{\sigma}_1(\bar{\sigma}_2|\bar{\tau})^{n-1}\bar{\sigma}_2 \parallel C$ and $\vec{v}{\approx}\vec{x}\bar{\sigma}_1(\bar{\sigma}_2|\bar{\tau})^{n-1}\bar{\tau} \parallel C$ are implied constrained clauses, and hence also $\vec{v}{\approx}\vec{x}\bar{\sigma}_1(\bar{\sigma}_2|\bar{\tau})^{n-1}(\bar{\sigma}_2|\bar{\tau}) \parallel C$ is implied, which is what we wanted to prove.

If the melting inference is of the shape

$$\frac{\vec{v}{\approx}\vec{x}\bar{\sigma}_1 \parallel C \quad \vec{v}{\approx}\vec{x}\bar{\sigma}_1\bar{\tau}' \parallel C'}{\vec{v}{\approx}\vec{x}\bar{\sigma}_1\bar{\tau}^* \parallel C}$$

the argumentation is analogous, except that we do not have to re-derive the suffix $\bar{\sigma}_2$. □

Concerning completeness, we can make use of the following proposition, which is an instance of Theorem 3.3:

Proposition 3.11 (Completeness)**.** *Let N be a finite existential query problem, let N^* be a finite saturation of N with respect to ordered query resolution, and let $\vec{v}{\approx}\vec{x}\bar{\sigma}_1 \parallel \Box, \ldots, \vec{v}{\approx}\vec{x}\bar{\sigma}_m \parallel \Box$ be the constrained clauses in N^* with empty clausal part. Then N has a Herbrand model iff $\bar{\sigma}_1|\ldots|\bar{\sigma}_m$ is not covering.*

As the ordered query resolution rule alone is already complete, the same holds for the combination of ordered query resolution and melting.

Hence, provided saturation terminates, we can express the initial problem whether $N \models_{Ind} \forall\vec{x}.\exists\vec{y}.A_1 \wedge \ldots \wedge A_n$ in terms of a coverage problem:

Corollary 3.12. *Let N be a satisfiable set of unconstrained Horn clauses, let N^* be a finite saturation of the set $N \cup \{\vec{v}\approx\vec{x} \,\|\, A_1, \ldots, A_n \rightarrow\}$ wrt. QRM, and let $\vec{v}\approx\vec{x}\bar{\sigma}_1 \,\|\, \Box, \ldots, \vec{v}\approx\vec{x}\bar{\sigma}_m \,\|\, \Box$ be the set of constrained clauses in N^* with empty clausal part. Then the following are equivalent:*

(1) $N \models_{Ind} \forall\vec{x}.\exists\vec{y}.A_1 \wedge \ldots \wedge A_n$

(2) $\bar{\sigma}_1 | \ldots | \bar{\sigma}_m$ *is not covering.*

3.4 Termination

We now show that, if a clause set $N \cup \{A_1, \ldots, A_n \rightarrow\}$ can be finitely saturated by ordered resolution (modulo variants and tautologies), then the calculus QRM allows us to finitely saturate the set $N \cup \{\vec{v}\approx\vec{x} \,\|\, A_1, \ldots, A_n \rightarrow\}$. Termination also ensures that derivations are *fair*, i.e. that every possible inference between derived constrained clauses will finally be redundant.

The derivation strategy that ensures termination proceeds in two stages. First a set of constrained clauses is derived using ordered query resolution only, then (the constraints of) these clauses are updated by meltings in such a way that the resulting clause set is saturated:

Strategy 3.13. (1) Perform inferences by ordered query resolution on the set $N \cup \{\vec{v}\approx\vec{x} \,\|\, A_1, \ldots, A_n \rightarrow\}$ according to the strategy that saturates $N \cup \{A_1, \ldots, A_n \rightarrow\}$. Whenever a meltable constrained clause is derived, do not consider this clause for any further ordered query resolution inferences.

(2) When no more inferences as in (1) are possible, start melting: Perform a (non-redundant) melting and *update* the derivation: Repeat, starting from the melted clause, all previous (non-redundant) resolution inferences that have the base clause as an ancestor and all (non-redundant) meltings that are possible with the newly derived constrained clauses. This is called an *elementary update*. Afterwards continue recursively with updates for the repeated meltings.

When the update is finished, reiterate this stage with another melting.

We can view an update as a generalization of the constraints of previously derived constrained clauses. When this happens, the "old" constrained clauses with their more specific constraints become redundant. This means that they can and will be ignored for the rest of the derivation and are effectively *replaced* by their more general counterparts.

Lemma 3.14. *If constrained clauses in a QRM derivation are arranged in a graph defined by the direct ancestor relation, then this graph is a forest (a set of trees).*

Proof. Initially, each constrained clause forms its own tree. These initial constrained clauses are and remain roots of the forest.

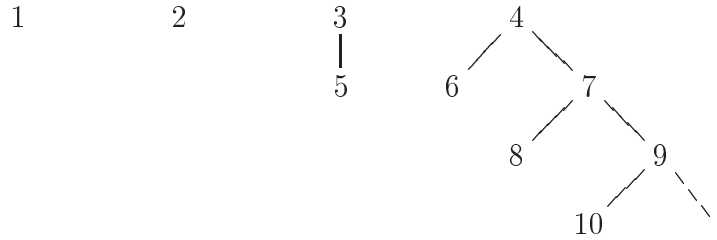
Recall that the ancestors of a constrained clause do not form a binary tree but a line, because we basically always ignore one premise.

Because of this, each inference by ordered query resolution extends the tree of the rightmost premise by one fresh leaf.

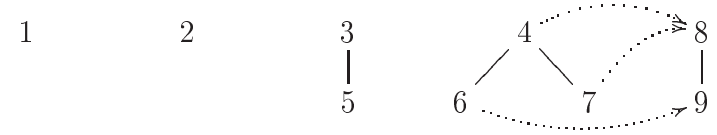
In melting inferences, the conclusion becomes a sibling of the base clause, i.e. it either becomes a fresh leaf in the tree of the base clause or, if the base clause is a root, it forms a new root of its own.

In any case, no inference can connect existing trees or introduce loops. □

Example 3.15. For Example 3.5, the graph looks as follows, where clauses are represented by their numbers:



For Example 3.7, which in fact follows Strategy 3.13, it looks as follows. The dotted lines are not part of the graph but illustrate replacements caused by the melting of clauses 4 and 7 into clause 8 and the following update:



Lemma 3.16 (Termination of Updates)**.** *The update following a melting inference step terminates.*

Proof. We consider the case of a melting as follows:

$$\frac{\vec{v}{\approx}\vec{x}\bar{\alpha}\bar{\sigma}^* \parallel C \quad \vec{v}{\approx}\vec{x}\bar{\alpha}\bar{\sigma}^*\bar{\tau}' \parallel C'}{\vec{v}{\approx}\vec{x}\bar{\alpha}(\bar{\sigma}|\bar{\tau})^* \parallel C}$$

Each elementary update is terminating, since there are only finitely many inferences to repeat. We show that the number of elementary updates in an update is finite and proceed by induction over the depth of the base clause in the ancestor-based forest.

Let

$$\frac{\vec{v}{\approx}\vec{x}\bar{\beta} \parallel E \quad \vec{v}{\approx}\vec{x}\bar{\beta}\bar{\rho}' \parallel E'}{\vec{v}{\approx}\vec{x}\bar{\beta}_1(\bar{\beta}_2|\bar{\rho})^* \parallel E}$$

be a melting inference that is redundant before the current elementary update. There are several possible cases, depending on whether and where one of the premises of the initial melting appears as an ancestor of $\vec{v}{\approx}\vec{x}\bar{\beta}\bar{\rho}' \parallel E'$:

- $\vec{v}{\approx}\vec{x}\bar{\alpha}\bar{\sigma}^* \parallel C$ is not an ancestor of $\vec{v}{\approx}\vec{x}\bar{\beta}\bar{\rho} \parallel E$. Then this melting is not affected by the elementary update.

- $E = C$. Then also $\bar{\beta} = \bar{\alpha}\bar{\sigma}^*$, and the first elementary update leads to a melting candidate

  $$\frac{\vec{v}{\approx}\vec{x}\bar{\alpha}(\bar{\sigma}|\bar{\tau})^* \parallel C \quad \vec{v}{\approx}\vec{x}\bar{\alpha}(\bar{\sigma}|\bar{\tau})^*\bar{\rho} \parallel C}{\vec{v}{\approx}\vec{x}\bar{\beta}(\bar{\sigma}|\bar{\tau}|\bar{\rho})^* \parallel C}$$

  Since the original melting was redundant before, $[\![\bar{\rho}]\!] \subseteq [\![\bar{\sigma}]\!]$. So the new melting candidate is also redundant and this branch of the update stops before the melting.

- $\vec{v}{\approx}\vec{x}\bar{\alpha}\bar{\sigma}^* \parallel C$ is an ancestor of $\vec{v}{\approx}\vec{x}\bar{\beta} \parallel E$, but not vice versa. Then $\bar{\beta} = \bar{\alpha}\bar{\sigma}^*\bar{\pi}$ and the elementary update leads to a melting candidate

  $$\frac{\vec{v}{\approx}\vec{x}\bar{\alpha}(\bar{\sigma}|\bar{\tau})^*\bar{\pi} \parallel E \quad \vec{v}{\approx}\vec{x}\bar{\alpha}(\bar{\sigma}|\bar{\tau})^*\bar{\pi}\bar{\rho} \parallel E}{\vec{v}{\approx}\vec{x}\bar{\alpha}(\bar{\sigma}|\bar{\tau})^*\bar{\pi}_1(\pi_2|\bar{\rho})^* \parallel E}$$

  where $\bar{\pi} = \bar{\pi}_1\bar{\pi}_2^*$, Since the original melting was redundant before, $[\![\bar{\rho}]\!] \subseteq [\![\bar{\pi}_2]\!]$. So the new melting candidate is also redundant and this branch of the update stops before the melting.

- $\vec{v}{\approx}\vec{x}\bar{\beta} \parallel E$ is an ancestor of $\vec{v}{\approx}\vec{x}\bar{\alpha}\bar{\sigma}^* \parallel C$. Then this melting has a base that lies strictly above the base of the originally inspected melting in the ancestor-based clause forest. Because of Lemma 3.14, we may inductively assume that the update initiated by the melting of $\vec{v}{\approx}\vec{x}\bar{\beta} \parallel E$ terminates.

This shows that the number of elementary updates is finite in this case. The case of a melting

$$\frac{\vec{v}{\approx}\vec{x}\bar{\alpha} \,\|\, C \quad \vec{v}{\approx}\vec{x}\bar{\alpha}\bar{\tau}' \,\|\, C'}{\vec{v}{\approx}\vec{x}\bar{\alpha}\bar{\tau}^* \,\|\, C}$$

works similarly. □

Lemma 3.17 (Saturation)**.** *Let $N \cup \{A_1, \ldots, A_n \to\}$ be a finite set of Horn clauses. If a QRM derivation that uses Strategy 3.13 and starts from the clause set $N \cup \{\vec{v}{\approx}\vec{x} \,\|\, A_1, \ldots, A_n \to\}$ terminates with a set N^*, then N^* is saturated with respect to QRM.*

Proof. We have to show that all melting and ordered query resolution inferences with premises in N^* are redundant.

All possible melting inferences with premises in N^* are redundant because by assumption, the second stage of the derivation terminates.

For any given melting inference, redundancy can have one of three reasons, all of which imply that the leftmost premise of the melting is redundant: If the conclusion is redundant, then so are both premises. If the rightmost premise is redundant, then so is the leftmost, because it was derived from a redundant constrained clause. Or finally, the leftmost premise itself might be redundant.

All in all, each meltable constrained clause in N^* is redundant, which means that also all ordered query resolution inferences with a meltable constrained clause are redundant.

The same holds for all ordered query resolution inferences with premises that are not meltable: They are redundant by construction after stage (1) and this can easily be seen to be maintained by each elementary update.

So all QRM inferences with premises in N^* are redundant, i.e. N^* is saturated. □

Theorem 3.18 (Termination)**.** *Let $N \cup \{A_1, \ldots, A_n \to\}$ be a finite set of Horn clauses that can be finitely saturated by ordered resolution. Then QRM finitely saturates the constrained clause set $N \cup \{\vec{v}{\approx}\vec{x} \,\|\, A_1, \ldots, A_n \to\}$ using Strategy 3.13.*

Proof. Because we already know from Lemma 3.17 that the strategy yields a saturated set, it remains to show the termination of the QRM derivation starting from $N \cup \{\vec{v}{\approx}\vec{x} \,\|\, A_1, \ldots, A_n \to\}$. We do so by proving the termination of each stage:

(1) Let M be the (possibly infinite) set obtained after the first stage of the QRM derivation starting from $N \cup \{\vec{v}{\approx}\vec{x} \,\|\, A_1, \ldots, A_n \to\}$. We first

show that M is finite. To do so, we show that the ancestor-based forest G for M is finite, i.e. it has only finitely many roots, is of finite depth, and is finitely branching.

- Only the finitely many constrained clauses in the initial set $N \cup \{\vec{v}\approx\vec{x} \parallel A_1, \ldots, A_n \rightarrow\}$ can appear as roots.
- Let N_0^* be a finite saturation of $N \cup \{A_1, \ldots, A_n \rightarrow\}$ with respect to ordered resolution.

  Consider two clauses $\vec{v}\approx\vec{x}\bar{\sigma} \parallel C$ and $\vec{v}\approx\vec{x}\bar{\sigma}\bar{\tau} \parallel C'$ on a common branch, such that none of them is a leaf. Because of our strategy of forbidding ordered query resolution inferences with clauses that can be melted with an ancestor, the two clauses cannot be melted. So there is no substitution expression $\bar{\tau}'$ such that $\vec{v}\approx\vec{x}\bar{\sigma}\bar{\tau} \parallel C'$ is a variant of $\vec{v}\approx\vec{x}\bar{\sigma}\bar{\tau}' \parallel C$.

  This means that every branch in G contains only a finite number of inner nodes for each element C of N_0^*: There is at most one for each subset of $\mathrm{vars}(C)$, since these correspond to the possibilities for $\mathrm{VRan}(\bar{\sigma}\bar{\tau}) \cap \mathrm{vars}(C)$ and hence to constrained C clauses that cannot be melted.

  So the graph is of finite depth.
- Finally, each node in G has a finite arity, as only finitely many inferences into the constrained clause at this node have been possible (namely from some of the finitely many unconstrained clauses only).

Since G is finite, and the same holds for M.

(2) It remains to show that the second stage terminates.

The number of meltable constrained clauses is finite when stage (1) is left, because M is finite. There are only finitely many iterations of stage (2), because each such iteration decreases the number of unreplaced meltable constrained clauses by one: The initial melting replaces both its premises (and in particular one meltable and unreplaced constrained clause), and each inference step in an update keeps the number of unreplaced meltable constrained clauses constant.

Moreover, we have proved in Lemma 3.16 that each update, i.e. each iteration, terminates.

So the whole second stage terminates.

□

4 Generalized Substitutions as Clause Sets

To make use of the completeness result Corollary 3.12, we are confronted with the task of deciding whether a finite disjunction $\bar{\sigma}_1|\ldots|\bar{\sigma}_n$ of substitution expressions is covering.

For a finite disjunction $\sigma_1|\ldots|\sigma_n$ of *basic* substitutions, this can be done using a completion procedure by Comon and Nieuwenhuis [8] that is based on a quantifier elimination procedure [6]. To do so, the problem is considered as a so-called disunification problem and reduced to an emptiness problem that is trivially decidable.

For the general case of substitution expressions, we follow a related approach. Our goal is to define a predicate $P_{\bar{\sigma}}$ for a substitution expression $\bar{\sigma}$ by a Horn clause set $N_{\bar{\sigma}}$ in such a way that a ground atom $P_{\bar{\sigma}}(\vec{s})$ holds in the minimal model of $N_{\bar{\sigma}}$ iff $\vec{s}$ lies in $\vec{x}[\![\bar{\sigma}]\!]$, where $\vec{x}$ is the domain of $\bar{\sigma}$. A clause $P_{\bar{\sigma}}(\vec{s}) \to P_{\bar{\tau}}(\vec{t})$ will mean that if a ground instance $\vec{s}\rho$ is a ground instance of an element of $\vec{x}[\![\bar{\sigma}]\!]$, then every ground instance of $\vec{t}\rho$ is a ground instance of an element of $\vec{y}[\![\bar{\tau}]\!]$, where $\vec{x}, \vec{y}$ are the respective domains.

In particular, this means that $\bar{\sigma}$ is covering iff $P_{\bar{\sigma}}$ is the total relation in the minimal model of $N_{\bar{\sigma}}$, i.e. iff $N_{\bar{\sigma}} \models_{Ind} \forall\vec{x}.P_{\bar{\sigma}}(\vec{x})$.

The above-mentioned completion procedure allows us to generate a Horn clause set $\check{N}_{\bar{\sigma}}$ for the complement of $P_{\bar{\sigma}}$, named $\check{P}_{\bar{\sigma}}$, such that $P_{\bar{\sigma}}$ is total in the minimal model of $N_{\bar{\sigma}}$ iff $\check{P}_{\bar{\sigma}}$ is empty in the minimal model of $\check{N}_{\bar{\sigma}}$. The completion procedure ensures that $\check{N}_{\bar{\sigma}}$ has only one Herbrand model over the given signature, which means that emptiness of $\check{P}_{\bar{\sigma}}$ in the minimal model of $\check{N}_{\bar{\sigma}}$ is equivalent to the emptiness of $\check{P}_{\bar{\sigma}}$ in every Herbrand model of $\check{N}_{\bar{\sigma}}$. Since $\check{N}_{\bar{\sigma}}$ is composed of unconstrained clauses, the problem can even be considered as a first-order problem that can be decided by ordered resolution (of unconstrained clauses).[1]

[1] This is, e.g., a direct consequence of [12, Proposition 4].

We will start the discussion by establishing the link between substitution expressions and clauses. Afterwards, we will shortly recall the quantifier elimination procedure (Section 4.2), explain how it is used for predicate completion, and finally show that it allows us to decide the coverage of substitution expressions (Section 4.3).

Definition 4.1 (Substitutions and Predicates)**.** Given a substitution expression $\bar{\sigma}$, we assign a predicate $P_{\bar{\tau}}$ to every occurrence of a substitution expression $\bar{\tau}$ that is a subexpression of $\bar{\sigma}$. If $\bar{\tau}$ is a basic substitution or disjunction or loop, $P_{\bar{\tau}}$ is a fresh predicate of arity $|\operatorname{dom}(\bar{\tau})|$. If $\bar{\tau} = \bar{\tau}_1\bar{\tau}_2$ is a concatenation, we define $P_{\bar{\tau}} = P_{\bar{\tau}_1}$.

Note that predicates are assigned not to substitution expressions but to occurrences of substitution expressions. For example in a substitution expression like $\{x \mapsto x\} \circ \{x \mapsto x\}$, two *different* predicates are assigned to the two occurrences of $\{x \mapsto x\}$.

Definition 4.2 (Substitutions and Clauses)**.** We translate substitution expressions $\bar{\sigma}$ to clause sets $N^0_{\bar{\sigma}}$ (which is just an intermediate representation) and $N_{\bar{\sigma}}$ as follows. Let P_{glue} be a fresh predicate of arity 0. This predicate will be used as a means to glue together the sets corresponding to different (occurrences of) substitution expressions. We assume an ordering $<$ on the set X of universal variables and write $\vec{x} = \operatorname{dom}(\bar{\sigma})$ if $\operatorname{dom}(\bar{\sigma}) = \{x_1, \ldots, x_n\}$ and $x_1 < \ldots < x_n$.

Expressions of the form $N[B/A]$ denote textual replacement of every occurrence of the atom A in the clause set N by the atom B.

$$
\begin{aligned}
N^0_{\sigma} &= \{P_{\text{glue}} \to P_{\sigma}(\vec{x}\sigma)\} \text{ where } \vec{x} = \operatorname{dom}(\sigma) \\
N^0_{\bar{\sigma}\bar{\tau}} &= N^0_{\bar{\tau}} \cup N^0_{\bar{\sigma}}[P_{\bar{\tau}}(\operatorname{dom}(\bar{\tau}))/P_{\text{glue}}] \\
N^0_{\bar{\sigma}^*} &= \{P_{\text{glue}} \to P_{\bar{\sigma}^*}(\operatorname{dom}(\bar{\sigma}))\} \\
&\quad \cup N^0_{\bar{\sigma}}[P_{\bar{\sigma}^*}(\operatorname{dom}(\bar{\sigma}))/P_{\text{glue}}] \\
&\quad \cup \{P_{\bar{\sigma}}(\operatorname{dom}(\bar{\sigma})) \to P_{\bar{\sigma}^*}(\operatorname{dom}(\bar{\sigma}))\} \\
N^0_{\bar{\sigma}_1|\bar{\sigma}_2} &= N^0_{\bar{\sigma}_1} \cup \{P_{\bar{\sigma}_1}(\vec{x}_1) \to P_{\bar{\sigma}_1|\bar{\sigma}_2}(\vec{y})\} \\
&\quad \cup N^0_{\bar{\sigma}_2} \cup \{P_{\bar{\sigma}_2}(\vec{x}_2) \to P_{\bar{\sigma}_1|\bar{\sigma}_2}(\vec{y})\} \\
&\quad \text{where } \vec{x}_i = \operatorname{dom}(\bar{\sigma}_i) \text{ and } \vec{y} = \operatorname{dom}(\bar{\sigma}_1|\bar{\sigma}_2)
\end{aligned}
$$

The set $N_{\bar{\sigma}}$ arises from $N^0_{\bar{\sigma}}$ by deletion of all occurrences of P_{glue}:

$$\begin{aligned}
N_\sigma &= \{ \to P_\sigma(\vec{x}\sigma)\} \text{ where } \vec{x} = \text{dom}(\sigma) \\
N_{\bar{\sigma}\bar{\tau}} &= N_{\bar{\tau}} \cup N^0_{\bar{\sigma}}[P_{\bar{\tau}}(\text{dom}(\bar{\tau}))/P_{\text{glue}}] \\
N_{\bar{\sigma}^*} &= \{ \to P_{\bar{\sigma}^*}(\text{dom}(\bar{\sigma}))\} \\
&\quad \cup N^0_{\bar{\sigma}}[P_{\bar{\sigma}^*}(\text{dom}(\bar{\sigma}))/P_{\text{glue}}] \\
&\quad \cup \{P_{\bar{\sigma}}(\text{dom}(\bar{\sigma})) \to P_{\bar{\sigma}^*}(\text{dom}(\bar{\sigma}))\} \\
N_{\bar{\sigma}_1|\bar{\sigma}_2} &= N_{\bar{\sigma}_1} \cup \{P_{\bar{\sigma}_1}(\vec{x}_1) \to P_{\bar{\sigma}_1|\bar{\sigma}_2}(\vec{y})\} \\
&\quad \cup N_{\bar{\sigma}_2} \cup \{P_{\bar{\sigma}_2}(\vec{x}_2) \to P_{\bar{\sigma}_1|\bar{\sigma}_2}(\vec{y})\} \\
&\quad \text{where } \vec{x}_i = \text{dom}(\bar{\sigma}_i) \text{ and } \vec{y} = \text{dom}(\bar{\sigma}_1|\bar{\sigma}_2)
\end{aligned}$$

Example 4.3. We consider the two basic substitutions $\sigma = \{x \mapsto s(x)\}$ and $\tau = \{x \mapsto 0\}$ appearing in Example 3.7 as parts of the substitution expression $\sigma^*\tau$. Then $N^0_\tau = \{P_{\text{glue}} \to P_\tau(0)\}$, and $N^0_\sigma = \{P_{\text{glue}} \to P_\sigma(s(x))\}$. Hence

$$\begin{aligned}
N^0_{\sigma^*} = \{ \quad & P_{\text{glue}} \to P_{\sigma^*}(x) \quad , \\
& P_{\sigma^*}(x) \to P_\sigma(s(x)) \quad , \\
& P_\sigma(x) \to P_{\sigma^*}(x) \quad \}
\end{aligned}$$

and finally (note that $P_{\sigma^*\tau} = P_{\sigma^*}$ as defined in Definition 4.1)

$$\begin{aligned}
N^0_{\sigma^*\tau} = \{ \quad & P_{\text{glue}} \to P_\tau(0), \\
& P_\tau(x) \to P_{\sigma^*\tau}(x) \quad , \\
& P_{\sigma^*\tau}(x) \to P_\sigma(s(x)) \quad , \\
& P_\sigma(x) \to P_{\sigma^*\tau}(x) \quad \} \ .
\end{aligned}$$

As the predicate P_{glue} is only used to glue clause sets together, we can throw it away now and arrive at the final representation of $\sigma^*\tau$ as a clause set:

$$\begin{aligned}
N_{\sigma^*\tau} = \{ \quad & \to P_\tau(0), \\
& P_\tau(x) \to P_{\sigma^*\tau}(x) \quad , \\
& P_{\sigma^*\tau}(x) \to P_\sigma(s(x)) \quad , \\
& P_\sigma(x) \to P_{\sigma^*\tau}(x) \quad \}
\end{aligned}$$

$N_{\sigma^*\tau}$ builds terms in a bottom-up approach: The first clause creates the constant 0, the second clause enters the σ loop, and the last two clauses allow to repeatedly wrap applications of $s(\ )$ around a term.

4.1 Equivalence of Substitution Expressions and Clause Sets

We will now show that each clause set $N_{\bar{\sigma}}$ describes exactly the (instances of the) term tuples generated by the respective substitution expression $\bar{\sigma}$,

provided that this substitution stems from a suitable derivation.

Definition 4.4. Let $\vec{v}{\approx}\vec{x}\bar{\sigma}_1, \ldots, \vec{v}{\approx}\vec{x}\bar{\sigma}_n$ be finitely many constraints appearing in a derivation starting from an existential query problem. Then $\bar{\sigma}_1 | \ldots | \bar{\sigma}_n$ is called a *derivation substitution.*

Such substitution expressions are especially well-behaved. For example, if disjunctions $\bar{\sigma}_1 | \bar{\sigma}_2$ appear, then $\bar{\sigma}_1$ and $\bar{\sigma}_2$ share the same domain.

Lemma 4.5. *Let $\bar{\sigma}$ be a derivation substitution. Then* $\mathrm{VRan}(\bar{\tau}_1) = \mathrm{dom}(\bar{\tau}_2)$ *for all subexpressions $\bar{\tau}_1 \circ \bar{\tau}_2$.*

Proof. This is true initially and can be easily seen to be maintained during each inference step. □

Lemma 4.6. *Let $\bar{\sigma}$ be a derivation substitution. Then* $\mathrm{dom}(\bar{\tau}_1) = \mathrm{dom}(\bar{\tau}_2)$ *for all subexpressions $\bar{\tau}_1 | \bar{\tau}_2$.*

Proof. If $\bar{\sigma} = \bar{\sigma}_1 | \ldots | \bar{\sigma}_n$ was created from $\vec{v}{\approx}\vec{x}\bar{\sigma}_1, \ldots, \vec{v}{\approx}\vec{x}\bar{\sigma}_n$, then the domains of all $\bar{\sigma}_i$ are identical by definition. A disjunction $\bar{\tau}_1 | \bar{\tau}_2$ inside $\bar{\sigma}_i$ can only have been introduced during a melting of type (3.i) involving two constraints $\vec{v}{\approx}\vec{x}\bar{\sigma}'\bar{\tau}_1^*$ and $\vec{v}{\approx}\vec{x}\bar{\sigma}'\bar{\tau}_1^*\bar{\tau}_2$. Hence $\mathrm{dom}(\bar{\tau}_2) = \mathrm{VRan}(\bar{\tau}_1^*) = \mathrm{dom}(\bar{\tau}_1)$ by Lemma 4.5 and the definition of VRan. □

Lemma 4.7. *Let $\bar{\sigma}$ be a substitution expression. In each clause in $N^0_{\bar{\sigma}}$ of the form $P_{glue} \to P(t_1, \ldots, t_n)$, the literal $P(t_1, \ldots, t_n)$ contains all variables in* $\mathrm{VRan}(\bar{\sigma})$.

Proof. Follows directly from the definitions. □

The clauses in $N^0_{\bar{\sigma}}$ and $N_{\bar{\sigma}}$ are particularly simple. On the one hand, all terms appearing on the left hand side are variables, on the other hand, we will now show that the clauses are universally reductive, i.e. all these variables also occur in the head, a property that is necessary for the applicability of the completion procedure we will use later on.

Proposition 4.8 (Universal Reductiveness of $N_{\bar{\sigma}}$)**.** *Let $\bar{\sigma}$ be a derivation substitution. Then $N_{\bar{\sigma}}$ is universally reductive.*

Proof. If $N^0_{\bar{\sigma}}$ is universally reductive, then obviously so is $N_{\bar{\sigma}}$. Hence it suffices to show that $N^0_{\bar{\sigma}}$ is universally reductive. We proceed by induction on the derivation substitution.

The clause $P_{\mathrm{glue}} \to P_\sigma(x_1\sigma, \ldots, x_n\sigma)$ for a single substitution σ is universally reductive because no variables occur outside the head.

In a conjunction $\bar{\sigma}\bar{\tau}$, we use that, inductively, $N^0_{\bar{\sigma}}$ and $N^0_{\bar{\tau}}$ are universally reductive. Moreover, $\mathrm{VRan}(\bar{\sigma}) = \mathrm{dom}(\bar{\tau})$ by Lemma 4.5, so the clause set $N^0_{\bar{\sigma}}[P_{\bar{\tau}}(\mathrm{dom}(\bar{\tau}))/P_{\mathrm{glue}}]$ is also universally reductive by Lemma 4.7.

The same argument applies to $N^0_{\bar{\sigma}^*}$, except that in this case $\mathrm{VRan}(\bar{\sigma}) = \mathrm{dom}(\bar{\sigma})$ by definition of VRan.

For $\bar{\sigma}_1|\bar{\sigma}_2$ we use that, inductively, both $N^0_{\bar{\sigma}_1}$ and $N^0_{\bar{\sigma}_2}$ are universally reductive. Moreover, since $\mathrm{dom}(\bar{\sigma}_1), \mathrm{dom}(\bar{\sigma}_2) \subseteq \mathrm{dom}(\bar{\sigma}_1|\bar{\sigma}_2)$, the newly introduced clauses are also universally reductive. □

In the following propositions, we write $N \vdash A$ if the atom A can be derived from the clause set N by a finite number of resolution steps.

Lemma 4.9 (Rule Lifting)**.** *Let $\bar{\sigma}$ be a derivation substitution. Let M be a clause set that does not contain any predicates of $N_{\bar{\sigma}}$, and let P be a predicate appearing in M. For terms $\vec{s}$, the following are equivalent:*

(1) $N^0_{\bar{\sigma}}[P(\vec{y})/P_{glue}] \cup M \vdash P_{\bar{\sigma}}(\vec{s})$.

(2) There are terms $\vec{s}_1$, $\vec{r}$, such that $\vec{s} = \vec{s}_1\{y_1 \mapsto r_1, \ldots, y_n \mapsto r_n\}$ and $N_{\bar{\sigma}} \vdash P_{\bar{\sigma}}(\vec{s}_1)$ and $M \vdash P(\vec{r})$.

This lemma formalizes a generalization of the idea that, if derivation substitutions are properly described by the corresponding clause sets, e.g. every atom $P_{\sigma_1}(\vec{s})$ that can be derived by $R_{\bar{\sigma}_1\bar{\sigma}_2}$ should correspond to an element of $[\![\bar{\sigma}_1\bar{\sigma}_2]\!]$ and hence be composed as $P_{\sigma_1}(\vec{s}) = P_{\sigma_1}(\vec{x}\sigma_1\sigma_2)$ with $\sigma_i \in [\![\bar{\sigma}_i]\!]$, such that $P_{\sigma_1}(\vec{x}\bar{\sigma}_1)$ can be derived by $R_{\bar{\sigma}_1}$ and $P_{\sigma_2}(\vec{y}\bar{\sigma}_2)$ can be derived by $R_{\bar{\sigma}_2}$.

Proposition 4.10 (Equivalence of Derivation Substitutions and Clauses)**.** *Let $\bar{\sigma}$ be a derivation substitution. Then $N_{\bar{\sigma}} \vdash P_{\bar{\sigma}}(\vec{t})$ iff $\vec{t} \in \vec{x}[\![\bar{\sigma}]\!]$.*

We prove Lemma 4.9 and Proposition 4.10 in parallel. The proofs rely on each other, however they do so in a well-founded way: The proof of Proposition 4.10 for a given substitution $\bar{\sigma}$ uses Lemma 4.9 only for strict subexpressions of $\bar{\sigma}$, and the proof of Lemma 4.9 for $\bar{\sigma}$ uses Proposition 4.10 for (possibly non-strict) subexpressions of $\bar{\sigma}$. In both proofs, we use existential quantifiers on the meta-level to denote the existence of terms with a given property.

Proof of Lemma 4.9. We proceed by induction over the structure of the substitution expression. Throughout this proof, we will abbreviate the application of a substitution $\{y_1 \mapsto r_1, \ldots, y_n \mapsto r_n\}$ to a term tuple $\vec{s}$ as $\vec{s}\{\vec{y} \mapsto \vec{r}\}$.

Consider a basic substitution σ. If $\{P(\vec{y}) \to P_\sigma(\vec{x}\sigma)\} \cup M \vdash P_\sigma(\vec{s})$, then the last inference in the derivation must have used the clause $P(\vec{y}) \to P_\sigma(\vec{x}\bar{\sigma})$,

i.e. $M \vdash P(\vec{r})$ for some $\vec{r}$ and s is of the form $s = s_1\{\vec{y} \mapsto \vec{r}\}$ for $s_1 = \vec{x}\sigma$. Quite obviously, $\{\to P_\sigma(\vec{x}\sigma)\} \vdash P_\sigma(\vec{s}_1)$.

For the other direction assume that $\{\to P_\sigma(\vec{x}\sigma)\} \vdash P_\sigma(\vec{s}_1)$ and $M \vdash P(\vec{r})$. Then clearly $\{P(\vec{y}) \to P_\sigma(\vec{x}\sigma)\} \cup M \vdash P_\sigma(\vec{s}_1)\{\vec{y} \mapsto \vec{r}\}$

In the case of a concatenation $\bar{\sigma}\bar{\tau}$ with $\mathrm{VRan}(\bar{\sigma}) = \mathrm{dom}(\bar{\tau}) = \vec{z}$ (cf. Lemma 4.5), we make excessive use of the induction hypotheses (i.h.):

$$
\begin{aligned}
&\exists \vec{s}_1, \vec{r}.\ N_{\bar{\sigma}\bar{\tau}} \vdash P_{\bar{\sigma}}(\vec{s}_1) \text{ and } M \vdash P(\vec{r}) \text{ and } \vec{s} = \vec{s}_1\{\vec{y} \mapsto \vec{r}\} \\
\iff &\exists \vec{s}_1, \vec{r}.\ \vec{s} = \vec{s}_1\{\vec{y} \mapsto \vec{r}\} \\
&\quad \text{and } N^0_{\bar{\sigma}}[P_{\bar{\tau}}(\vec{z})/P_{\mathrm{glue}}] \cup N_{\bar{\tau}} \vdash P_{\bar{\sigma}}(\vec{s}_1) \text{ and } M \vdash P(\vec{r}) \\
\overset{\text{i.h.}}{\iff} &\exists \vec{s}_1, \vec{r}, \vec{s}_2, \vec{t}.\ \vec{s} = \vec{s}_1\{\vec{y} \mapsto \vec{r}\} \text{ and } \vec{s}_1 = \vec{s}_2\{\vec{z} \mapsto \vec{t}\} \\
&\quad \text{and } N_{\bar{\sigma}} \vdash P_{\bar{\sigma}}(\vec{s}_2) \text{ and } N_{\bar{\tau}} \vdash P_{\bar{\tau}}(\vec{t}) \text{ and } M \vdash P(\vec{r}) \\
\overset{\text{i.h.}}{\iff} &\exists \vec{s}_1, \vec{r}, \vec{s}_2, \vec{t}.\ \vec{s} = \vec{s}_1\{\vec{y} \mapsto \vec{r}\} \text{ and } \vec{s}_1 = \vec{s}_2\{\vec{z} \mapsto \vec{t}\} \\
&\quad \text{and } N_{\bar{\sigma}} \vdash P_{\bar{\sigma}}(\vec{s}_2) \text{ and } N_{\bar{\tau}}[P(\vec{y})/P_{\mathrm{glue}}] \cup M \vdash P_{\bar{\tau}}(\vec{t}\{\vec{y} \mapsto \vec{r}\}) \\
\overset{\text{i.h.}}{\iff} &N^0_{\bar{\sigma}}[P_{\bar{\tau}}(\vec{z})/P_{\mathrm{glue}}] \cup N_{\bar{\tau}}[P(\vec{y})/P_{\mathrm{glue}}] \cup M \vdash P_{\bar{\sigma}}(\vec{s}) \\
\iff &N^0_{\bar{\sigma}\bar{\tau}}[P(\vec{y})/P_{\mathrm{glue}}] \cup M \vdash P_{\bar{\sigma}}(\vec{s})
\end{aligned}
$$

In the case of a loop $\bar{\sigma}^*$, let $I(\vec{t}_1)$ be the minimal number of times the clause $P_{\bar{\sigma}}(\vec{x}) \to P_{\bar{\sigma}^*}(\vec{x}) \in N_{\bar{\sigma}^*}$ is needed in a derivation $N_{\bar{\sigma}^*} \vdash P_{\bar{\sigma}^*}(\vec{t}_1)$, and let $J(\vec{t}_1)$ be the minimal number of times this clause is needed in a derivation $N^0_{\bar{\sigma}^*}[P(\vec{y})/P_{\mathrm{glue}}] \cup M \vdash P_{\bar{\sigma}^*}(\vec{t}_1)$. If there is no such derivation, then $I(\vec{t}_1)$ or $J(\vec{t}_1)$ is undefined.

We proceed by induction over $I(\vec{t}_1)$, showing that for all n, that $J(\vec{t}) = n$ and $N^0_{\bar{\sigma}^*}[P(\vec{y})/P_{\mathrm{glue}}] \cup M \vdash P_{\bar{\sigma}^*}(\vec{t})$ is equivalent to the existence of $\vec{t}_1$ and $\vec{r}$ s.th. $I(\vec{t}_1) = n$ and $N_{\bar{\sigma}^*} \vdash P_{\bar{\sigma}^*}(\vec{t}_1)$ and $M \vdash P(\vec{r})$.

For $I(t) = 0$:

$$
\begin{aligned}
&\exists \vec{t}_1, \vec{r}.\ \vec{t} = \vec{t}_1\{\vec{x} \mapsto \vec{r}\} \text{ and } N_{\bar{\sigma}^*} \vdash P_{\bar{\sigma}^*}(\vec{t}_1) \text{ and } M \vdash P(\vec{r}) \text{ and } I(\vec{t}_1) = 0 \\
\iff &\exists \vec{t}_1, \vec{r}.\ \vec{t} = \vec{t}_1\{\vec{x} \mapsto \vec{r}\} \\
&\quad \text{and } N^0_{\bar{\sigma}}[P_{\bar{\sigma}^*}(\vec{x})/P_{\mathrm{glue}}] \cup \{P_{\bar{\sigma}^*}(\vec{x})\} \vdash P_{\bar{\sigma}^*}(\vec{t}_1) \text{ and } M \vdash P(\vec{r}) \\
\iff &\exists \vec{t}_1, \vec{r}.\ \vec{t} = \vec{t}_1\{\vec{x} \mapsto \vec{r}\} \text{ and } \{P_{\bar{\sigma}^*}(\vec{x})\} \vdash P_{\bar{\sigma}^*}(\vec{t}_1) \text{ and } M \vdash P(\vec{r}) \\
\overset{\text{i.h.}}{\iff} &\{P(\vec{y}) \to P_{\bar{\sigma}^*}(\vec{x})\} \cup M \vdash P_{\bar{\sigma}^*}(\vec{t}) \\
\iff &N^0_{\bar{\sigma}^*}[P(\vec{y})/P_{\mathrm{glue}}] \cup M \vdash P_{\bar{\sigma}^*}(\vec{t}) \text{ and } J(\vec{t}) = 0
\end{aligned}
$$

For $I(t) > 0$, let $\bar{\sigma}'$ be a copy of $\bar{\sigma}$, such that $N_{\bar{\sigma}}$ and $N_{\bar{\sigma}'}$ do not share any predicate symbols.

From Proposition 4.10, we know that $N_{\bar{\sigma}^*} \vdash P_{\bar{\sigma}^*}(\vec{t}_1)$ iff $N_{\bar{\sigma}^*} \vdash P_{\bar{\sigma}^*}(\vec{t}_1)$, and that $N_{\bar{\sigma}^*\bar{\sigma}'}[P(\vec{y})/P_{\mathrm{glue}}] \cup M \vdash P_{\bar{\sigma}^*}(\vec{t})$ iff $N_{\bar{\sigma}^*\bar{\sigma}'}[P(\vec{y})/P_{\mathrm{glue}}] \cup M \vdash P_{\bar{\sigma}^*}(\vec{t})$.

The use of Proposition 4.10 for $\bar{\sigma}^*$ is sound, since the proof of 4.10 uses the current lemma only for strict sub-expressions (in this case: $\bar{\sigma}$).

$$
\begin{aligned}
& \exists \vec{t_1}, \vec{r}.\ \vec{t} = \vec{t_1}\{\vec{x} \mapsto \vec{r}\} \text{ and } I(\vec{t_1}) = n+1 \\
& \qquad \text{and } N_{\bar{\sigma}^*} \vdash P_{\bar{\sigma}^*}(\vec{t_1}) \text{ and } M \vdash P(\vec{r}) \\
\iff & \exists \vec{t_1}, \vec{r}.\ \vec{t} = \vec{t_1}\{\vec{x} \mapsto \vec{r}\} \text{ and } I(\vec{t_1}) = n+1 \\
& \qquad \text{and } N_{\bar{\sigma}^*\bar{\sigma}'} \vdash P_{\bar{\sigma}^*}(\vec{t_1}) \text{ and } M \vdash P(\vec{r}) \\
\iff & \exists \vec{t_1}, \vec{r}.\ \vec{t} = \vec{t_1}\{\vec{x} \mapsto \vec{r}\} \text{ and } I(\vec{t_1}) = n+1 \\
& \qquad N^0_{\bar{\sigma}^*}[P_{\bar{\sigma}'}(\vec{x})/P_{\text{glue}}] \cup N_{\bar{\sigma}'} \vdash P_{\bar{\sigma}^*}(\vec{t_1}) \text{ and } M \vdash P(\vec{r}) \\
\overset{\text{i.h.}}{\iff} & \exists \vec{t_1}, \vec{r}, \vec{s}_1, \vec{t}_2.\ \vec{t} = \vec{t_1}\{\vec{x} \mapsto \vec{r}\} \text{ and } \vec{t_1} = \vec{t_2}\{\vec{x} \mapsto \vec{s}_1\} \text{ and } I(\vec{t_2}) = n \\
& \qquad \text{and } N_{\bar{\sigma}^*} \vdash P_{\bar{\sigma}^*}(\vec{t_2}) \text{ and } N_{\bar{\sigma}'} \vdash P_{\bar{\sigma}'}(\vec{s}_1) \text{ and } M \vdash P(\vec{r}) \\
\overset{\text{i.h.}}{\iff} & \exists \vec{t_1}, \vec{r}, \vec{s}_1, \vec{t}_2.\ \vec{t} = \vec{t_1}\{\vec{x} \mapsto \vec{r}\} \text{ and } \vec{t_1} = \vec{t_2}\{\vec{x} \mapsto \vec{s}_1\} \text{ and } I(\vec{t_2}) = n \\
& \qquad \text{and } N_{\bar{\sigma}^*} \vdash P_{\bar{\sigma}^*}(\vec{t_2}) \\
& \qquad \text{and } N^0_{\bar{\sigma}'}[P(\vec{y})/P_{\text{glue}}] \cup M \vdash P_{\bar{\sigma}'}(\vec{s}_1\{\vec{x} \mapsto \vec{r}\}) \\
\overset{\text{i.h.}}{\iff} & \exists \vec{t_1}, \vec{r}, \vec{s}_1, \vec{t}_2.\ \vec{t} = \vec{t_1}\{\vec{x} \mapsto \vec{r}\} \text{ and } \vec{t_1} = \vec{t_2}\{\vec{x} \mapsto \vec{s}_1\} \text{ and } J(\vec{t_2}) = n \\
& \qquad \text{and } N^0_{\bar{\sigma}^*}[P_{\bar{\sigma}'}(\vec{x})/P_{\text{glue}}] \cup N^0_{\bar{\sigma}'}[P(\vec{y})/P_{\text{glue}}] \cup M \\
& \qquad\quad \vdash P_{\bar{\sigma}^*}(\vec{t_2}\{\vec{x} \mapsto \vec{s}_1\{\vec{x} \mapsto \vec{r}\}\}) \\
\overset{\text{i.h.}}{\iff} & \exists \vec{t_1}, \vec{r}, \vec{s}_1, \vec{t}_2.\ \vec{t} = \vec{t_1}\{\vec{x} \mapsto \vec{r}\} \text{ and } \vec{t_1} = \vec{t_2}\{\vec{x} \mapsto \vec{s}_1\} \text{ and } J(\vec{t_2}) = n \\
& \qquad \text{and } N^0_{\bar{\sigma}^*\bar{\sigma}'}[P(\vec{y})/P_{\text{glue}}] \cup M \vdash P_{\bar{\sigma}^*}(\vec{t_2}\{\vec{x} \mapsto \vec{s}_1\{\vec{x} \mapsto \vec{r}\}\}) \\
\iff & J(\vec{t}) = n+1 \text{ and } N^0_{\bar{\sigma}^*}[P(\vec{y})/P_{\text{glue}}] \cup M \vdash P_{\bar{\sigma}^*}(\vec{t})
\end{aligned}
$$

In the case of a disjunction $\bar{\sigma}_1|\bar{\sigma}_2$ with $\text{dom}(\bar{\sigma}_1) = \text{dom}(\bar{\sigma}_2) = \vec{x}$, we use

that $N_{\bar\sigma_1}$ and $N_{\bar\sigma_2}$ do not share any predicate symbols.

$$
\begin{aligned}
&\exists \vec{s}_1, \vec{r}.\ \vec{s} = \vec{s}_1\{\vec{y} \mapsto \vec{r}\} \text{ and } M \vdash P(\vec{r}) \text{ and } N_{\bar\sigma_1|\bar\sigma_2} \vdash P_{\bar\sigma_1|\bar\sigma_2}(\vec{s}_1) \\
\iff &\exists \vec{s}_1, \vec{r}.\ \exists i.\ \vec{s} = \vec{s}_1\{\vec{y} \mapsto \vec{r}\} \text{ and } M \vdash P(\vec{r}) \\
&\qquad \text{and } N_{\bar\sigma_i} \cup \{P_{\bar\sigma_i}(\vec{x}) \to P_{\bar\sigma_1|\bar\sigma_2}(\vec{x})\} \vdash P_{\bar\sigma_1|\bar\sigma_2}(\vec{s}_1) \\
\overset{\text{i.h.}}{\iff} &\exists \vec{s}_1, \vec{r}.\ \exists i.\ \vec{s} = \vec{s}_1\{\vec{y} \mapsto \vec{r}\} \text{ and } M \vdash P(\vec{r}) \text{ and } N_{\bar\sigma_i} \vdash P_{\bar\sigma_i}(\vec{s}_1) \\
\overset{\text{i.h.}}{\iff} &\exists i.\ N^0_{\bar\sigma_i}[P(\vec{y})/P_{\text{glue}}] \cup M \vdash P_{\bar\sigma_i}(\vec{s}) \\
\overset{\text{i.h.}}{\iff} &\exists i.\ N^0_{\bar\sigma_i}[P(\vec{y})/P_{\text{glue}}] \cup M \cup \{P_{\bar\sigma_i}(\vec{x}) \to P_{\bar\sigma_1|\bar\sigma_2}(\vec{x})\} \vdash P_{\bar\sigma_1|\bar\sigma_2}(\vec{s}) \\
\iff &N^0_{\bar\sigma_1}[P(\vec{y})/P_{\text{glue}}] \cup \{P_{\bar\sigma_1}(\vec{x}) \to P_{\bar\sigma_1|\bar\sigma_2}(\vec{x})\} \\
&\qquad \cup N^0_{\bar\sigma_2}[P(\vec{y})/P_{\text{glue}}] \cup \{P_{\bar\sigma_2}(\vec{x}) \to P_{\bar\sigma_1|\bar\sigma_2}(\vec{x})\}) \cup M \vdash P_{\bar\sigma_1|\bar\sigma_2}(\vec{s}) \\
\iff &N_{\bar\sigma_1|\bar\sigma_2}[P(\vec{y})/P_{\text{glue}}] \cup M \vdash P_{\bar\sigma_1|\bar\sigma_2}(\vec{s})
\end{aligned}
$$

This completes the proof. □

Proof of Proposition 4.10. The validity of the proposition is obvious for basic substitutions.

In the case of a concatenation $\bar\sigma\bar\tau$ with $\mathrm{dom}(\bar\sigma) = \vec{x}$ and $\mathrm{dom}(\bar\tau) = \vec{y}$, we have the following equality:

$$
\begin{aligned}
&\{\vec{s} \mid N_{\bar\sigma\bar\tau} \vdash P_{\bar\sigma}(\vec{s})\} \\
=\ &\left\{\vec{s} \mid N^0_{\bar\sigma}[P_{\bar\tau}(\vec{y})/P_{\text{glue}}] \cup N_{\bar\tau} \vdash P_{\bar\sigma}(\vec{s})\right\} \\
\overset{4.9}{=}\ &\{\vec{s} \mid \exists \vec{t}, \vec{s}_1.\ \vec{s} = \vec{s}_1\{\vec{y} \mapsto \vec{t}\} \\
&\qquad \text{and } N_{\bar\tau} \vdash P_{\bar\tau}(\vec{t}) \text{ and } N_{\bar\sigma} \vdash P_{\bar\sigma}(\vec{s}_1)\} \\
\overset{\text{i.h.}}{=}\ &\{\vec{s} \mid \exists \vec{t}, \vec{s}_1.\ \vec{s} = \vec{s}_1\{\vec{y} \mapsto \vec{t}\} \text{ and } \vec{t} \in \vec{y}[\![\bar\tau]\!] \text{ and } \vec{s}_1 \in \vec{x}[\![\bar\sigma]\!]\} \\
=\ &\vec{x}[\![\bar\sigma\bar\tau]\!]
\end{aligned}
$$

In the case of a loop $\bar\sigma^*$ with $\mathrm{dom}(\bar\sigma) = \vec{x}$, we show inductively that $\vec{x}[\![\bar\sigma^n]\!]$ is the subset of $\{\vec{s} \mid N_{\bar\sigma^*} \vdash P_{\bar\sigma^*}(\vec{s})\}$ of those term tuples $\vec{s}$ for which n is the minimal number of instances of the iteration clause $P_{\bar\sigma}(\vec{x}) \to P_{\bar\sigma^*}(\vec{x})$ needed in a derivation $N_{\bar\sigma^*} \vdash P_{\bar\sigma^*}(\vec{s})$, denoted $I(\vec{s}) = n$. For $n = 0$:

$$
\begin{aligned}
&\{\vec{s} \mid N_{\bar\sigma^*} \vdash P_{\bar\sigma^*}(\vec{s}) \text{ and } I(\vec{s}) = 0\} \\
=\ &\left\{\vec{s} \mid N^0_{\bar\sigma}[P_{\bar\sigma^*}(\vec{x})/P_{\text{glue}}] \cup \{\to P_{\bar\sigma^*}(\vec{x})\} \vdash P_{\bar\sigma^*}(\vec{s})\right\} \\
=\ &\{\vec{s} \mid \{\to P_{\bar\sigma^*}(\vec{x})\} \vdash P_{\bar\sigma^*}(\vec{s})\} \\
=\ &\vec{x}[\![\bar\sigma^0]\!]
\end{aligned}
$$

For $n > 0$, we can split a derivation $N_{\bar{\sigma}^*} \vdash P_{\bar{\sigma}^*}(\vec{s})$ at the point where the clause $P_{\bar{\sigma}}(\vec{x}) \to P_{\bar{\sigma}^*}(\vec{x})$ is used for the last time:

$$
\begin{aligned}
&\{\vec{s} \mid N_{\bar{\sigma}^*} \vdash P_{\bar{\sigma}^*}(\vec{s}) \text{ and } I(\vec{s}) = n\} \\
&= \{\vec{s} \mid \exists \vec{t}.\ N_{\bar{\sigma}^*} \vdash P_{\bar{\sigma}^*}(\vec{t}) \text{ and } I(\vec{t}) = n-1 \\
&\qquad \text{and } N^0_{\bar{\sigma}}[P_{\bar{\sigma}^*}(\vec{x})/P_{\text{glue}}] \cup \{\to P_{\bar{\sigma}^*}(\vec{t})\} \vdash P_{\bar{\sigma}}(\vec{s})\} \\
&\overset{\text{i.h.}}{=} \{\vec{s} \mid \exists \vec{t}.\ \vec{t} \in \vec{x}[\![\bar{\sigma}^{n-1}]\!] \\
&\qquad \text{and } N^0_{\bar{\sigma}}[P_{\bar{\sigma}^*}(\vec{x})/P_{\text{glue}}] \cup \{\to P_{\bar{\sigma}^*}(\vec{t})\} \vdash P_{\bar{\sigma}}(\vec{s})\} \\
&\overset{4.9}{=} \{\vec{s} \mid \exists \vec{t}, \vec{s}_1.\ \vec{s} = \vec{s}_1\{\vec{y} \mapsto \vec{t}\} \text{ and } \vec{t} \in \vec{x}[\![\bar{\sigma}^{n-1}]\!] \\
&\qquad \text{and } N_{\bar{\sigma}} \vdash P_{\bar{\sigma}}(\vec{s}_1)\} \\
&\overset{\text{i.h.}}{=} \{\vec{s} \mid \exists \vec{t}, \vec{s}_1.\ \vec{s} = \vec{s}_1\{\vec{y} \mapsto \vec{t}\} \text{ and } \vec{t} \in \vec{x}[\![\bar{\sigma}^{n-1}]\!] \\
&\qquad \text{and } \vec{s}_1 \in \vec{x}[\![\bar{\sigma}]\!]\} \\
&= \vec{x}[\![\bar{\sigma}^n]\!]
\end{aligned}
$$

Finally, in the case of a disjunction $\bar{\sigma}_1|\bar{\sigma}_2$ with domains $\text{dom}(\bar{\sigma}_1) = \text{dom}(\bar{\sigma}_2) = \vec{x}$ (cf. Lemma 4.6), we have the following equality, where we use that $N_{\bar{\sigma}_1}$ and $N_{\bar{\sigma}_2}$ do not share any predicate symbols:

$$
\begin{aligned}
&\left\{\vec{s} \mid N_{\bar{\sigma}_1|\bar{\sigma}_2} \vdash P_{\bar{\sigma}}(\vec{s})\right\} \\
&= \{\vec{s} \mid N_{\bar{\sigma}_1} \cup \{P_{\bar{\sigma}_1}(\vec{x}) \to P_{\bar{\sigma}}(\vec{x})\} \vdash P_{\bar{\sigma}}(\vec{s}) \\
&\qquad \text{or } N_{\bar{\sigma}_2} \cup \{P_{\bar{\sigma}_2}(\vec{x}) \to P_{\bar{\sigma}}(\vec{x})\} \vdash P_{\bar{\sigma}}(\vec{s})\} \\
&\overset{4.9}{=} \{\vec{s} \mid N_{\bar{\sigma}_1} \vdash P_{\bar{\sigma}_1}(\vec{s}) \text{ or } N_{\bar{\sigma}_2} \vdash P_{\bar{\sigma}_2}(\vec{s})\} \\
&\overset{\text{i.h.}}{=} \{\vec{s} \mid \vec{s} \in \vec{x}[\![\bar{\sigma}_1]\!] \text{ or } \vec{s} \in \vec{x}[\![\bar{\sigma}_2]\!]\} \\
&= \vec{x}[\![\bar{\sigma}_1]\!] \cup \vec{x}[\![\bar{\sigma}_2]\!] \\
&= \vec{x}[\![\bar{\sigma}_1|\bar{\sigma}_2]\!]
\end{aligned}
$$

This completes the proof. □

Since resolution is first-order complete, we can conclude that the terms entailed by $N_{\bar{\sigma}}$ are exactly those covered by $\bar{\sigma}$:

Corollary 4.11. *Let $\bar{\sigma}$ be a derivation substitution. Then $N_{\bar{\sigma}} \models P_{\bar{\sigma}}(\vec{t})$ iff $\bar{\sigma}$ is covering for $\{\vec{t}\}$, i.e. the set $\left\{\vec{t} \mid N_{\bar{\sigma}} \models P_{\bar{\sigma}}(\vec{t})\right\}$ is the maximal set for which $\bar{\sigma}$ is covering.*

4.2 The Quantifier Elimination Algorithm by Comon and Lescanne

Now that we know how to transform $\bar{\sigma}$ into an equivalent set $N_{\bar{\sigma}}$ of Horn clauses, we will concentrate on how to decide whether $P_{\bar{\sigma}}$ is the total relation in $N_{\bar{\sigma}}$.

The transformation of this problem into an emptiness problem will be presented in Section 4.3. It follows a procedure introduced by Comon and Nieuwenhuis [8] that is based on Clark's completion [5] and a quantifier elimination procedure by Comon and Lescanne [7].

In this section, we present the rules of the quantifier elimination algorithm as they are given, in a revised version, in [6]. To simplify the presentation, we omit constraints that are only relevant for the termination of the rule system.

The algorithm works on arbitrary formulas over a given finite signature $\Sigma = (\mathcal{P}, \mathcal{F})$ where all atoms are either (syntactic) equations, written $s \simeq t$, or predicative atoms over Σ, or one of the atoms "true" and "false".

In the rules, different designators stand for elements of restricted sets of variables or terms:

$x, x_i, \ldots$	free variables
$y, y_i, \ldots$	universally quantified variables
$v, v_i, \ldots$	existentially quantified variables
$z, z_i, \ldots$	free or existentially quantified variables
$w, w_i, \ldots$	any variables
$t, t_i, \ldots$	terms without any occurrence of a universally quantified variable
$s, s_i, u, u_i \ldots$	terms
$u[u'], \ldots$	the term u' is a subterm of the term u
$d, d_i, \ldots$	disjunctions of equations and disequations and predicative literals
$\phi, \phi_i, \psi, \psi_i, \ldots$	quantifier-free formulas

While it is unusual that three kinds of variables (free, universal, and existential) appear, this variety stems from the intended application of finding solutions, i.e. satisfying instantiations of the free variables, of a formula containing existential and universal quantifiers. Because of this, the free variables require a special treatment and cannot be regarded as bound by an additional quantifier on top of the formula.

The quantifier elimination rules are given below. They can be applied at any position of a formula.

Replacement:

R1: $z \simeq t \wedge \phi[z] \rightsquigarrow z \simeq t \wedge \phi\{z \mapsto t\}$
R2: $z \not\simeq t \vee \phi[z] \rightsquigarrow z \not\simeq t \vee \phi\{z \mapsto t\}$
If $z \notin \text{vars}(t)$.

Universal quantifier elimination:

UE1: $\forall \vec{y}, y.\phi \rightsquigarrow \forall \vec{y}.\phi$
UE2: $\phi \wedge y \not\simeq u \rightsquigarrow \text{false}$
UE3: $y \not\simeq u \vee d \rightsquigarrow d\{y \mapsto u\}$
UE5: $\phi \wedge (w_1 \simeq u_1 \vee \ldots \vee w_n \simeq u_n \vee d) \rightsquigarrow \phi \wedge d$

If $y \notin \text{vars}(\phi)$ (for UE1), $y \notin \text{vars}(u)$ (for UE2-3) and each $w_i \simeq u_i$ contains at least one occurrence of a universally quantified variable, but d does not (for UE5). UE5 is only applicable if $\mathcal{T}(\mathcal{F})$ is infinite.

Merging:

M1: $z \simeq t \wedge z \simeq u \rightsquigarrow z \simeq t \wedge t \simeq u$
M2: $w \not\simeq s \vee w \not\simeq u \rightsquigarrow w \not\simeq s \vee s \not\simeq u$
M3: $z \simeq t \wedge z \not\simeq u \rightsquigarrow z \simeq t \wedge t \not\simeq u$
M4: $w \simeq s \vee w \not\simeq u \rightsquigarrow s \simeq u \vee w \not\simeq u$
M5: $z \simeq t \wedge (z \not\simeq u \vee \phi) \rightsquigarrow z \simeq t \wedge (t \not\simeq u \vee \phi)$
M6: $z \simeq t \wedge (z \simeq u \vee \phi) \rightsquigarrow z \simeq t \wedge (t \simeq u \vee \phi)$

The merging rules are not necessary for termination or completeness.

Existential quantifier elimination:

EE1: $\exists v.\phi \rightsquigarrow \phi$
EE2: $\exists \vec{v}, v.v \simeq t \wedge \phi \rightsquigarrow \exists \vec{v}.\phi$
EE3: $\exists \vec{v}.\psi \wedge \phi \rightsquigarrow \exists \vec{v}.\phi$

If $v \notin \text{vars}(\phi, t)$ (for EE1-2) and $\psi = (d_1 \vee z_1 \not\simeq t_1) \wedge \ldots \wedge (d_n \vee z_n \not\simeq t_n)$ and there exists $v \in \vec{v} \cap \text{vars}(z_1, t_1) \cap \text{vars}(z_n, t_n)$ which does not occur in ϕ (for EE3).

Normalization:

A set of Boolean simplification rules, like $a \wedge a \rightsquigarrow a$.

Disjunction lifting:

DL: $\exists \vec{v}.\forall \vec{y}.\phi \wedge (\phi_1 \vee \phi_2) \rightsquigarrow (\exists \vec{v}.\forall \vec{y}.\phi \wedge \phi_1) \vee (\exists \vec{v}.\forall \vec{y}.\phi \wedge \phi_2)$

If $\text{vars}(\phi_1) \cap \vec{y} = \emptyset$ or $\text{vars}(\phi_2) \cap \vec{y} = \emptyset$.

Conflict:

C1: $f(u_1, \ldots, u_m) \simeq g(u'_1, \ldots, u'_n) \rightsquigarrow \text{false}$
C2: $f(u_1, \ldots, u_m) \not\simeq g(u'_1, \ldots, u'_n) \rightsquigarrow \text{true}$

If $f \neq g$.

Decomposition:

D1: $f(u_1, \ldots, u_n) \simeq f(u'_1, \ldots, u'_n) \rightsquigarrow u_1 \simeq u'_1 \wedge \ldots \wedge u_n \simeq u'_n$
D2: $f(u_1, \ldots, u_n) \not\simeq f(u'_1, \ldots, u'_n) \rightsquigarrow u_1 \not\simeq u'_1 \vee \ldots \vee u_n \not\simeq u'_n$

Occurrence check:

O1: $s \simeq u[s] \rightsquigarrow$ false

O2: $s \not\simeq u[s] \rightsquigarrow$ true

If $u \neq s$.

Explosion:

E: $\exists \vec{v}.\forall \vec{y}.\phi \rightsquigarrow \bigvee_{f \in \mathcal{F}} \exists \vec{v}_1, \vec{v}.\forall y.\phi \wedge z \simeq f(\vec{v}_1)$

If no other rule can be applied, $\vec{v} \cap \mathrm{vars}(\phi) = \emptyset$ and there exists in ϕ an equation $z \simeq u$ or disequation $z \not\simeq u$ where u contains an occurrence of a universally quantified variable.

This rule set allows to eliminate universal quantifiers from purely equational formulas of a certain shape:

Theorem 4.12 ([6, Theorem 9])**.** *Let $\psi = \bigvee_j (\exists \vec{w}_j.\forall \vec{y}_j.\phi_j)$ be a formula such that each ϕ_j is a quantifier-free formula over a finite signature $\Sigma = (\mathcal{P}, \mathcal{F})$.*

Then the above rule set transforms ψ in finitely many steps into a formula ψ' of the form

$$\psi' = \bigvee_j (\exists \vec{w}_j.x_{j1} \simeq t_{j1} \wedge \ldots \wedge x_{jn_j} \simeq t_{jn_j} \wedge z_{j1} \not\simeq u_{j1} \wedge \ldots \wedge z_{jm_j} \not\simeq u_{jm_j})$$

where the x_{ji} are variables occurring only once in each disjunct, and each z_{jk} is a variable that is not identical with u_{jk}.

Moreover, ψ and ψ' are satisfied by the same assignments of terms in $\mathcal{T}(\mathcal{F})$ to the free variables.

4.3 Predicate Completion

In detail, the completion procedure introduced by Comon and Nieuwenhuis works as follows:

(1) Let N be a finite and saturated set of Horn clauses over a finite signature $\Sigma = (\mathcal{P}, \mathcal{F})$. Combine all clauses with a common head predicate P into a single formula $\phi_P \to P(\vec{x})$ where

$$\phi_P = \exists \vec{y}. \bigvee_{\Gamma \to P(\vec{t}) \in N} (\vec{x} \simeq \vec{t} \wedge \bigwedge_{A \in \Gamma} A) \ ,$$

the y_i are the variables appearing in N, and the x_j are fresh variables.

(2) In the minimal model $N_{\mathcal{I}}$, this formula is equivalent to $\forall \vec{x}.(\neg \phi_P \to \neg P(\vec{x}))$. Under certain circumstances (cf. Lemma 4.13 below), $\neg \phi_P$ can be transformed using quantifier elimination (cf. Section 4.2) into an equivalent formula ψ_P that does not contain any universal quantifiers.

(3) After replacing each occurrence of a negated atom $\neg P(\vec{t})$ in ψ_P by $\check{P}(\vec{t})$, this formula can in turn be written as a (possibly equational) clause set $\check{R}_P$. The union of all R_P, $P \in \mathcal{P}$, is called the *completion* of N and denoted by $\check{N}$.

Lemma 4.13 ([8, Lemma 47])**.** *Let N be a set of universally reductive and non-equational unconstrained Horn clauses. Then the set $\check{N}$ computed by the completion procedure by Comon and Nieuwenhuis is a set of unconstrained clauses (possibly containing equations) defining the completion of the predicates in N.*

Since we have shown that $N_{\bar{\sigma}}$ is universally reductive if $\bar{\sigma}$ is a derivation substitution (Proposition 4.8), we know:

Lemma 4.14. *Let $\bar{\sigma}$ be a derivation substitution. The completion procedure by Comon and Nieuwenhuis computes from $N_{\bar{\sigma}}$ a clause set $\check{N}_{\bar{\sigma}}$ defining the completion of the predicates in $N_{\bar{\sigma}}$.*

In general, the result of the quantifier elimination procedure is a formula corresponding to a set of clauses of the form $\Gamma \to \check{P}(\vec{t}) \wedge E$, where E is a conjunction of equations. E.g., the clause $Q(s(x)) \to P(x, x, z)$ is transformed to $\check{Q}(s(x)) \to \check{P}(x, y, z), x \simeq y$.

When all appearing basic substitutions have a unary domain and are linear, however, the clauses in $\check{N}$ do not contain any equations, and they fall into a class for which the emptiness of $\check{N}$ is decidable by ordered resolution (Theorem 4.17).

Proposition 4.15. *Let $\bar{\sigma}$ be a derivation substitution such that all basic substitutions appearing in $\bar{\sigma}$ are linear. For the predicates in $N_{\bar{\sigma}}$, the algorithm by Comon and Nieuwenhuis computes clauses of the following types:*

(1) $\to \check{P}(\vec{t})$

(2) $\check{P}_1(\vec{x}) \to \check{P}(\vec{t})$

(3) $\check{P}_1(\vec{x}), \check{P}_2(\vec{x}) \to \check{P}(\vec{x})$

The positive literal of each computed clause is linear.

Proof. A substitution predicate P_σ is defined by a single clause that is either of the shape $\to P_\sigma(\vec{t})$ or $P_{\bar{\tau}}(\vec{x}) \to P_\sigma(\vec{t})$. Its completion consists of a finite set of linear clauses of the form $\to \check{P}_\sigma(\vec{s})$ describing the term tuples not covered by $\vec{t}$ and, in the second case, additionally the clause $\check{P}_{\bar{\tau}}(\vec{x}) \to \check{P}_\sigma(\vec{t})$.

A loop predicate $P_{\bar{\sigma}^*}$ is defined by $P_{\bar{\sigma}}(\vec{x}) \to P_{\bar{\sigma}^*}(\vec{x})$ and either $\to P_{\bar{\sigma}^*}(\vec{x})$ or $P_{\bar{\tau}}(\vec{x}) \to P_{\bar{\sigma}^*}(\vec{x})$. The completion is the empty set in the first case (as $P_{\bar{\sigma}^*}$ is total) and $\{\check{P}_{\bar{\sigma}}(\vec{x}), \check{P}_{\bar{\tau}}(\vec{x}) \to \check{P}_{\bar{\sigma}^*}(\vec{x})\}$ in the second case.

The two clauses defining a disjunction predicate $P_{\bar{\sigma}_1|\bar{\sigma}_2}$ are transformed into $\{\check{P}_{\bar{\sigma}_1}(\vec{x}), \check{P}_{\bar{\sigma}_2}(\vec{x}) \to \check{P}_{\bar{\sigma}_1|\bar{\sigma}_2}(\vec{x})\}$. □

The fact that all negative literals contain only variables is inherited from $N_{\bar{\sigma}}$. That no equations appear is due to the linearity of the positive literals in $N_{\bar{\sigma}}$.

Example 4.16. In Example 4.3, we had the following clause set $N_{\sigma^*\tau}$:

$$\begin{aligned} &\to P_\tau(0) \\ P_{\sigma^*\tau}(x) &\to P_\sigma(s(x)) \\ P_\tau(x) &\to P_{\sigma^*\tau}(x) \\ P_\sigma(x) &\to P_{\sigma^*\tau}(x) \end{aligned}$$

Hence $\check{N}_{\sigma^*\tau}$ consists of the following clauses:

$$\begin{aligned} &\to \check{P}_\tau(s(x)) \\ &\to \check{P}_\sigma(0) \\ \check{P}_{\sigma^*\tau}(x) &\to \check{P}_\sigma(s(x)) \\ \check{P}_\tau(x), \check{P}_\sigma(x) &\to \check{P}_{\sigma^*\tau}(x) \end{aligned}$$

In this example, all appearing predicates are monadic. When there is more than one existential variable or when the signature contains function symbols of arity at least two, predicates of higher arity appear also in $\check{N}$.

Theorem 4.17 (Decidability of Coverage)**.** *Let $\bar{\sigma}$ be a derivation substitution over a finite signature such that all basic substitutions in $\bar{\sigma}$ have a unary domain and are linear. It is decidable whether $\bar{\sigma}$ is covering.*

Proof. We translate $\bar{\sigma}$ into a clause set $N_{\bar{\sigma}}$. All predicates in $N_{\bar{\sigma}}$ are unary. The resulting clause set $\check{N}_{\bar{\sigma}}$ defining the completion of all appearing predicates again contains only clauses of the form $\check{P}_1(x), \ldots, \check{P}_n(x) \to \check{P}(t)$ (Proposition 4.15). Weidenbach [16] showed that such a clause set is equivalent to a so-called *sort theory*, a clause set in which additionally all clauses are shallow. For sort theories, emptiness is decidable by ordered resolution [16, 15]. Emptiness of $\check{P}_{\bar{\sigma}}$ in turn is equivalent to the coverage of the substitution $\bar{\sigma}$ (completion and Corollary 4.11). □

Example 4.18. We shortly present the final step of the proof using our running example. Here, all clauses in $\check{N}_{\sigma^*\tau}$ are already shallow (cf. Example 4.16). We choose an ordering $\succ$ on atoms such that $P_{\sigma^*\tau}(x) \succ P_\tau(x), P_\sigma(x)$ and $P_\sigma(s(x)) \succ P_{\sigma^*\tau}(x)$ and saturate $\check{N}_{\sigma^*\tau} \cup \{\check{P}_{\sigma^*\tau}(x) \to\}$ using ordered resolution. The derivation runs as follows:

$$
\begin{aligned}
\text{clauses in } \check{N}_{\sigma^*\tau}\text{: } 1 &: & &\to \check{P}_\tau(s(x)) \\
2 &: & &\to \check{P}_\sigma(0) \\
3 &: & \check{P}_{\sigma^*\tau}(x) &\to \check{P}_\sigma(s(x)) \\
4 &: & \check{P}_\tau(x), \check{P}_\sigma(x) &\to \check{P}_{\sigma^*\tau}(x) \\
\text{negated query: } 5 &: & \check{P}_{\sigma^*\tau}(x) &\to \\
\text{resolve}(4{,}5) = 6 &: & \check{P}_\tau(x), \check{P}_\sigma(x) &\to \\
\text{resolve}(1{,}6) = 7 &: & \check{P}_\sigma(s(x)) &\to \\
\text{resolve}(2{,}6) = 8 &: & \check{P}_\tau(0) &\to
\end{aligned}
$$

At this point, the clause set is saturated. It is consistent, so $\check{N}_{\sigma^*\tau} \not\models \exists x.\check{P}_{\sigma^*}(x)$. Since first-order and inductive validity coincide for Horn clause sets and positive existential queries, this implies that $\check{P}$ is empty in the minimal model of $\check{N}_{\sigma^*\tau}$. So $\sigma^*\tau$ is covering and $N \models_{Ind} C$.

5 Decidability of Inductive Validity

As a combination of our complete and terminating ordered resolution calculus QRM for constrained clauses and the completion-based treatment of the substitution expressions that can appear during saturation, we obtain the following decidability result:

Theorem 5.1. *Let $N \cup \{A_1, \ldots, A_n \to\}$ be a set of Horn clauses without equality over a finite signature Σ, where*

(1) all function symbols in Σ are at most unary,

(2) all positive literals in N are linear, and

(3) $N \cup \{A_1, \ldots, A_n \to\}$ belongs to a class that can be finitely saturated by ordered resolution.

Let $\{x, y_1, \ldots, y_m\}$ be the set of variables that appear in $A_1, \ldots, A_n$. It is decidable whether $N \models_{Ind} \forall x.\exists y_1, \ldots, y_m.(A_1 \wedge \ldots \wedge A_n)$.

Proof. By Corollary 3.12 and Theorem 3.18, the unconstrained clause set $N \cup \{v \approx x \parallel A_1, \ldots, A_n \to\}$ can be finitely saturated by ordered resolution with melting, such that the deduced constrained clauses $v \approx x\bar{\sigma}_1 \parallel \square, \ldots, v \approx x\bar{\sigma}_k \parallel \square$ with empty clausal part in the saturated set correspond to a substitution expression $\bar{\sigma} = \bar{\sigma}_1 | \ldots | \bar{\sigma}_k$ that is covering iff $N \models_{Ind} \forall x.\exists y_1, \ldots, y_m.(A_1 \wedge \ldots \wedge A_n)$.

Since the domain $\mathrm{dom}(\bar{\sigma}) = \{x\}$ of the derivation constraint $\bar{\sigma}$ contains only one element and Σ contains only unary function symbols, the domain of all basic substitutions appearing in $\bar{\sigma}$ has cardinality 1. These substitutions are also linear because all positive literals in N are linear and hence the most general unifiers appearing in each resolution step are linear.

Hence coverage of $\bar{\sigma}$ is decidable by Theorem 4.17. □

6 Conclusion

We have shown that the problem

$$N \models_{Ind} \forall\exists^*(A_1 \wedge \ldots \wedge A_n)$$

is decidable over a finite signature consisting of constants and unary function symbols if all positive literals in N are linear and $N \cup \{A_1, \ldots, A_n \rightarrow\}$ belongs to a class that can be finitely saturated by first-order ordered resolution (with variant subsumption and tautology deletion). Our proof is constructive and based on an ordered resolution calculus for constrained clauses $\vec{v} \approx \vec{x}\bar{\sigma} \parallel C$ and predicate completion.

Among the related work on automated inductive theorem proving, the approach most closely related to ours is the one by Comon and Nieuwenhuis [8]. Given a universally reductive Horn clause set N and a query $\forall^* C$, they use predicate completion to compute a so-called I-axiomatization A and reduce the initial problem to the satisfiability of $N \cup A \cup \{C\}$. This method is complete but not terminating. In fact, since A is a clause set over the original predicates P (and not over $\check{P}$), it is usually not Horn nor does the saturation of $N \cup A \cup \{C\}$ terminate.

Another general method based on saturation is the one by Ganzinger and Stuber [11]. Given a not necessarily Horn but universally reductive clause set N and a query $\forall^* C$, they basically saturate $N \cup \{C\}$. Even if $N \cup \{C\}$ saturates finitely, this results in a non-complete procedure. They also present a way to guarantee completeness, at the cost that the resulting algorithm almost never terminates. The reason for this is that their method usually results in an enumeration of all ground query instances.

A subtle, but important difference between both these methods and ours is that we add the negated query to N for saturation (while they add the query positively), which makes all derived clauses also hold in the minimal model.

Another intensely studied approach is via test sets [13, 2, 3]. Test set methods are complete for several classes of equational clauses or rewrite systems and universal queries. Termination results for these approaches usually

require strong properties like a terminating rewrite system on constructor terms.

Other approaches that provide decision procedures are those in the tradition of Caferra and Zabel [4] or Kapur [13, 9]. However, they are restricted to equational unit clauses. Related publications by Peltier [14] require that N has a unique Herbrand model (not only a unique minimal one), which leads back to I-Axiomatizations.

In summary, our approach is the first to yield a both terminating and complete algorithm to decide the inductive validity of queries that contain a $\forall\exists^*$ quantifier alternation.

Extensions of the approach might include a relaxation of its side conditions. Although both the superposition calculus of [12] and the completion procedure of [8] are also applicable to clauses containing equality literals, it is not obvious how to extend this treatment of equality also to clauses containing generalized substitutions. The main problem here is that term rewriting cannot easily be extended to the rewriting of generalized substitutions.

However, since both our resolution calculus and the completion procedure work equally well on $\forall^*\exists^*$ queries, on clauses over an arbitrary signature, and on clauses containing non-linear positive atoms, the reduction to an emptiness problem is also possible in these extended settings. The resulting set $\check{R}$ may contain both non-monadic predicates and equational atoms. It is a natural next step to explore under which conditions these extensions lead to predicates $\check{P}$ that are nevertheless defined in such a way that emptiness is still decidable.

Bibliography

[1] L. Bachmair and H. Ganzinger. Rewrite-based equational theorem proving with selection and simplification. *Journal of Logic and Computation*, 4(3):217–247, 1994. Revised version of Technical Report MPI-I-91-208, 1991.

[2] A. Bouhoula. Automated theorem proving by test set induction. *Journal of Symbolic Computation*, 23(1):47–77, 1997.

[3] A. Bouhoula and J.-P. Jouannaud. Automata-driven automated induction. In *LICS*, pages 14–25, 1997.

[4] R. Caferra and N. Zabel. A method for simultaneous search for refutations and models by equational constraint solving. *Journal of Symbolic Computation*, 13(6):613–642, 1992.

[5] K. L. Clark. Negation as failure. In *Logic and Data Bases*, pages 293–322. Plenum Press, 1977.

[6] H. Comon. Disunification: A survey. In J.-L. Lassez and G. Plotkin, editors, *Computational Logic: Essays in Honor of Alan Robinson*, pages 322–359. MIT Press, Cambridge, MA, 1991.

[7] H. Comon and P. Lescanne. Equational problems and disunification. *Journal of Symbolic Computation*, 7(3-4):371–425, 1989.

[8] H. Comon and R. Nieuwenhuis. Induction = I-axiomatization + first-order consistency. *Information and Computation*, 159(1/2):151–186, May 2000.

[9] S. Falke and D. Kapur. Inductive decidability using implicit induction. In M. Hermann and A. Voronkov, editors, *Proceedings of the 13th International Conference on Logic for Programming, Artificial Intelligence*

and Reasoning (LPAR '06), Phnom Penh, Cambodia, volume 4246 of *Lecture Notes in Artificial Intelligence*, pages 45–59. Springer, 2006.

[10] H. Ganzinger and H. D. Nivelle. A superposition decision procedure for the guarded fragment with equality. In *Proc. 14th IEEE Symposium on Logic in Computer Science*, pages 295–305. IEEE Computer Society Press, 1999.

[11] H. Ganzinger and J. Stuber. Inductive theorem proving by consistency for first-order clauses. In J. Buchmann, H. Ganzinger, and W. Paul, editors, *Informatik - Festschrift zum 60. Geburtstag von Günter Hotz*, pages 441–462. Teubner, 1992. Also in Proc. CTRS'92, LNCS 656, pp. 226–241.

[12] M. Horbach and C. Weidenbach. Superposition for fixed domains. In M. Kaminski and S. Martini, editors, *Proc. of the 17th Annual Conference of the European Association for Computer Science Logic, CSL 2008*, volume 5213 of *Lecture Notes in Computer Science*, pages 293–307, Berlin / Heidelberg, September 2008. Springer.

[13] D. Kapur, P. Narendran, and H. Zhang. Automating inductionless induction using test sets. *Journal of Symbolic Computation*, 11(1/2):81–111, 1991.

[14] N. Peltier. Model building with ordered resolution: extracting models from saturated clause sets. *Journal of Symbolic Computation*, 36(1-2):5–48, 2003.

[15] H. Seidl and K. N. Verma. Flat and one-variable clauses: Complexity of verifying cryptographic protocols with single blind copying. In *Logic for Programming, Artificial Intelligence, and Reasoning, 11th International Conference, LPAR 2004, Montevideo, Uruguay, March 14-18, 2005, Proceedings*, volume 3452 of *LNCS*, pages 79–94. Springer, 2004.

[16] C. Weidenbach. Towards an automatic analysis of security protocols in first-order logic. In H. Ganzinger, editor, *16th International Conference on Automated Deduction, CADE-16*, volume 1632 of *LNAI*, pages 378–382. Springer, 1999.

[17] C. Weidenbach. Combining superposition, sorts and splitting. In A. Robinson and A. Voronkov, editors, *Handbook of Automated Reasoning*, volume 2, chapter 27, pages 1965–2012. Elsevier, 2001.

Below you find a list of the most recent technical reports of the Max-Planck-Institut für Informatik. They are available via WWW using the URL `http://www.mpi-inf.mpg.de`. If you have any questions concerning WWW access, please contact `reports@mpi-inf.mpg.de`. Paper copies (which are not necessarily free of charge) can be ordered either by regular mail or by e-mail at the address below.

Max-Planck-Institut für Informatik
Library
attn. Anja Becker
Stuhlsatzenhausweg 85
66123 Saarbrücken
GERMANY
e-mail: `library@mpi-inf.mpg.de`

MPI-I-2009-RG1-002	P. Wischnewski, C. Weidenbach	Contextual rewriting
MPI-I-2009-5-006	S. Bedathur, K. Berberich, J. Dittrich, N. Mamoulis, G. Weikum	Scalable phrase mining for ad-hoc text analytics
MPI-I-2009-5-004	N. Preda, F.M. Suchanek, G. Kasneci, T. Neumann, G. Weikum	Coupling knowledge bases and web services for active knowledge
MPI-I-2009-5-003	T. Neumann, G. Weikum	The RDF-3X engine for scalable management of RDF data
MPI-I-2008-RG1-001	A. Fietzke, C. Weidenbach	Labelled splitting
MPI-I-2008-5-004	F. Suchanek, M. Sozio, G. Weikum	SOFI: a self-organizing framework for information extraction
MPI-I-2008-5-003	F.M. Suchanek, G. de Melo, A. Pease	Integrating Yago into the suggested upper merged ontology
MPI-I-2008-5-002	T. Neumann, G. Moerkotte	Single phase construction of optimal DAG-structured QEPs
MPI-I-2008-5-001	F. Suchanek, G. Kasneci, M. Ramanath, M. Sozio, G. Weikum	STAR: Steiner tree approximation in relationship-graphs
MPI-I-2008-4-003	T. Schultz, H. Theisel, H. Seidel	Crease surfaces: from theory to extraction and application to diffusion tensor MRI
MPI-I-2008-4-002	W. Saleem, D. Wang, A. Belyaev, H. Seidel	Estimating complexity of 3D shapes using view similarity
MPI-I-2008-1-001	D. Ajwani, I. Malinger, U. Meyer, S. Toledo	Characterizing the performance of Flash memory storage devices and its impact on algorithm design
MPI-I-2007-RG1-002	T. Hillenbrand, C. Weidenbach	Superposition for finite domains
MPI-I-2007-5-003	F.M. Suchanek, G. Kasneci, G. Weikum	Yago : a large ontology from Wikipedia and WordNet
MPI-I-2007-5-002	K. Berberich, S. Bedathur, T. Neumann, G. Weikum	A time machine for text search
MPI-I-2007-5-001	G. Kasneci, F.M. Suchanek, G. Ifrim, M. Ramanath, G. Weikum	NAGA: searching and ranking knowledge
MPI-I-2007-4-008	J. Gall, T. Brox, B. Rosenhahn, H. Seidel	Global stochastic optimization for robust and accurate human motion capture
MPI-I-2007-4-007	R. Herzog, V. Havran, K. Myszkowski, H. Seidel	Global illumination using photon ray splatting
MPI-I-2007-4-006	C. Dyken, G. Ziegler, C. Theobalt, H. Seidel	GPU marching cubes on shader model 3.0 and 4.0
MPI-I-2007-4-005	T. Schultz, J. Weickert, H. Seidel	A higher-order structure tensor
MPI-I-2007-4-004	C. Stoll, E. de Aguiar, C. Theobalt, H. Seidel	A volumetric approach to interactive shape editing
MPI-I-2007-4-003	R. Bargmann, V. Blanz, H. Seidel	A nonlinear viseme model for triphone-based speech synthesis
MPI-I-2007-4-002	T. Langer, H. Seidel	Construction of smooth maps with mean value coordinates
MPI-I-2007-4-001	J. Gall, B. Rosenhahn, H. Seidel	Clustered stochastic optimization for object recognition and pose estimation

MPI-I-2007-2-001	A. Podelski, S. Wagner	A method and a tool for automatic veriication of region stability for hybrid systems
MPI-I-2007-1-003	A. Gidenstam, M. Papatriantafilou	LFthreads: a lock-free thread library
MPI-I-2007-1-002	E. Althaus, S. Canzar	A Lagrangian relaxation approach for the multiple sequence alignment problem
MPI-I-2007-1-001	E. Berberich, L. Kettner	Linear-time reordering in a sweep-line algorithm for algebraic curves intersecting in a common point
MPI-I-2006-5-006	G. Kasnec, F.M. Suchanek, G. Weikum	Yago - a core of semantic knowledge
MPI-I-2006-5-005	R. Angelova, S. Siersdorfer	A neighborhood-based approach for clustering of linked document collections
MPI-I-2006-5-004	F. Suchanek, G. Ifrim, G. Weikum	Combining linguistic and statistical analysis to extract relations from web documents
MPI-I-2006-5-003	V. Scholz, M. Magnor	Garment texture editing in monocular video sequences based on color-coded printing patterns
MPI-I-2006-5-002	H. Bast, D. Majumdar, R. Schenkel, M. Theobald, G. Weikum	IO-Top-k: index-access optimized top-k query processing
MPI-I-2006-5-001	M. Bender, S. Michel, G. Weikum, P. Triantafilou	Overlap-aware global df estimation in distributed information retrieval systems
MPI-I-2006-4-010	A. Belyaev, T. Langer, H. Seidel	Mean value coordinates for arbitrary spherical polygons and polyhedra in $\mathbb{R}^3$
MPI-I-2006-4-009	J. Gall, J. Potthoff, B. Rosenhahn, C. Schnoerr, H. Seidel	Interacting and annealing particle filters: mathematics and a recipe for applications
MPI-I-2006-4-008	I. Albrecht, M. Kipp, M. Neff, H. Seidel	Gesture modeling and animation by imitation
MPI-I-2006-4-007	O. Schall, A. Belyaev, H. Seidel	Feature-preserving non-local denoising of static and time-varying range data
MPI-I-2006-4-006	C. Theobalt, N. Ahmed, H. Lensch, M. Magnor, H. Seidel	Enhanced dynamic reflectometry for relightable free-viewpoint video
MPI-I-2006-4-005	A. Belyaev, H. Seidel, S. Yoshizawa	Skeleton-driven laplacian mesh deformations
MPI-I-2006-4-004	V. Havran, R. Herzog, H. Seidel	On fast construction of spatial hierarchies for ray tracing
MPI-I-2006-4-003	E. de Aguiar, R. Zayer, C. Theobalt, M. Magnor, H. Seidel	A framework for natural animation of digitized models
MPI-I-2006-4-002	G. Ziegler, A. Tevs, C. Theobalt, H. Seidel	GPU point list generation through histogram pyramids
MPI-I-2006-4-001	A. Efremov, R. Mantiuk, K. Myszkowski, H. Seidel	Design and evaluation of backward compatible high dynamic range video compression
MPI-I-2006-2-001	T. Wies, V. Kuncak, K. Zee, A. Podelski, M. Rinard	On verifying complex properties using symbolic shape analysis
MPI-I-2006-1-007	H. Bast, I. Weber, C.W. Mortensen	Output-sensitive autocompletion search
MPI-I-2006-1-006	M. Kerber	Division-free computation of subresultants using bezout matrices
MPI-I-2006-1-005	A. Eigenwillig, L. Kettner, N. Wolpert	Snap rounding of Bézier curves
MPI-I-2006-1-004	S. Funke, S. Laue, R. Naujoks, L. Zvi	Power assignment problems in wireless communication
MPI-I-2005-5-002	S. Siersdorfer, G. Weikum	Automated retraining methods for document classification and their parameter tuning
MPI-I-2005-4-006	C. Fuchs, M. Goesele, T. Chen, H. Seidel	An emperical model for heterogeneous translucent objects
MPI-I-2005-4-005	G. Krawczyk, M. Goesele, H. Seidel	Photometric calibration of high dynamic range cameras
MPI-I-2005-4-004	C. Theobalt, N. Ahmed, E. De Aguiar, G. Ziegler, H. Lensch, M.A. Magnor, H. Seidel	Joint motion and reflectance capture for creating relightable 3D videos
MPI-I-2005-4-003	T. Langer, A.G. Belyaev, H. Seidel	Analysis and design of discrete normals and curvatures
MPI-I-2005-4-002	O. Schall, A. Belyaev, H. Seidel	Sparse meshing of uncertain and noisy surface scattered data