UPMAIL Technical Report No. 118
October 30, 1995

A Note on Semantics of Logic Programs with Equality Based on Complete Sets of E-Unifiers

Anatoli Degtyarev*
Andrei Voronkov†

Computing Science Department
Uppsala University
Box 311, S-751 05 Uppsala,
Sweden

email {anatoli,voronkov}@csd.uu.se

*Supported by a grant from the Swedish Royal Academy of Sciences
†Supported by a TFR grant

Abstract

We discuss semantics of equational Horn-clause programs based on the notion of a complete set of E-unifiers. We prove incompleteness of SLDE†-resolution introduced by Gallier and Raatz. We also define and compare several fixpoint semantics.

Section 1

Introduction

The notions of an E-unifier and a complete set of E-unifiers were introduced in the area of automated theorem proving with equality [17]. They generalize the notions of a unifier and a most general unifier, respectively, for the case of built-in equational theories. In logic programming, E-unifiers were introduced in [12]. The complete sets of unifiers in logic programming have been considered in [8, 5, 6, 10, 18, 19]. All these papers except [5, 6] considered restricted classes of logic programs with equality.

Despite different formulations, these restricted programs can be characterized as pairs $(\mathcal{P}, E)$, where E is an equational theory (a set of equations) and $\mathcal{P}$ is a logic program where equality can only occur in the bodies of the clauses. Definite programs of this kind were called *well-behaved programs* in [6].

E-unifiers and equational Horn-clause programs without any restrictions were considered in [6]. The model-theoretic semantics of such clauses is well known since long ago [1, 15]. However, there is no uniform generally accepted procedural interpretation. Article [6] defined two procedural interpretations for equational Horn-clause programs: SLDE-resolution and SLDE†-resolution. SLDE-resolution is similar to unrestricted SLD-resolution [14], but with E-unifiers instead of unifiers. SLDE†-resolution is similar to SLD-resolution, but with complete sets of E-unifiers instead of most general unifiers.

In [6] it was proved that SLDE-resolution is complete. The completeness of SLDE†-resolution was established only for well-behaved programs. The completeness of SLDE†-resolution in general was left as an open problem.

In this article we prove that SLDE†-resolution is incomplete. We also consider some properties of the complete sets of E-unifiers related to the fixpoint semantics of logic programs. We discuss several possible ways of defining the immediate consequence operator based on the complete sets of E-unifiers. We show that for some natural definition the immediate consequence operator is not monotone and the least Herbrand model of the program is not a fixpoint of the operator. We also define several immediate consequence operators having all desirable properties.

Section 2

Preliminaries

We assume the knowledge of the standard notions of substitutions and unification, including variants and instances. Substitutions are denoted by $\{t_1/x_1, \ldots, t_n/x_n\}$. The empty substitution is denoted by ε. All function symbols belong to a fixed signature Σ. Constants are considered as function symbols of arity 0. We assume that all terms use variables of a fixed countable set $\mathcal{Y}$. Variables are denoted by x, y, z, v, maybe with indices. The set of all terms of the signature Σ is denoted by $\mathcal{T}_\Sigma$. An *equation* is any expression of the form $s = t$, where $s, t \in \mathcal{T}_\Sigma$. We denote by $s \approx t$ that the terms s and t are identical. Notation $\rightleftharpoons$ means "equal by definition". The symbol $\vdash$ is used for the derivability in first-order logic with equality. By derivability of a formula φ from a set of formulas Φ we understand the derivability of the universal closure $\forall\varphi$ from the universal closures of formulas in Φ. For example, we have $x = f(x) \vdash y = f(f(y))$.

The set of all variables occurring in a term t is denoted by $var(t)$. The *domain* and the *range* of a substitution θ are sets of variables defined by

$$\begin{array}{l} dom(\theta) \rightleftharpoons \{x \mid x\theta \not\approx x\} \\ ran(\theta) \rightleftharpoons \{x \mid \text{there is } y \in dom(\theta) \text{ such that } x \in var(y\theta)\}, \end{array}$$

respectively. Let t be a term with one occurrence of a variable x. We write $t[x]$ to denote this occurrence of x. Then $t[s]$ denotes the term $t\{s/x\}$.

Let s, t, r_1, r_2 be terms and θ be a substitution. Then we write $r_1 \Leftrightarrow_{[s=t,\theta]} r_2$, iff

1. $dom(\theta) \subseteq var(s) \bigcup var(t)$;
2. there is a term r such that $r_1 \approx r[s\theta]$ and $r_2 \approx r[t\theta]$, or $r_2 \approx r[s\theta]$ and $r_1 \approx r[t\theta]$.

The following result is used in the definition of SLDE†-resolution in Section 3:

Lemma 2.1 Suppose $E \vdash s = t$. There are equations $s_1 = t_1, \ldots, s_n = t_n$ in E and substitutions $\theta_1, \ldots, \theta_n$ such that

$$s \Leftrightarrow_{[s_1=t_1,\theta_1]} \ldots \Leftrightarrow_{[s_n=t_n,\theta_n]} t$$

The following definition is a simplified version of a definition of [6] for the case when all atoms are equations.

Definition 2.1 An *equational Horn-clause program* is any set of *Horn clauses* of the form $s_1 = t_1 \wedge \ldots \wedge s_n = t_n \supset s = t$, denoted $s = t$ `:-` $s_1 = t_1, \ldots, s_n = t_n$. A *goal* is any formula of the form $\neg(s_1 = t_1 \wedge \ldots \wedge s_n = t_n)$, denoted `:-` $s_1 = t_1, \ldots, s_n = t_n$. The *empty goal* is denoted by $\Box$.

Unlike the functional logic programming approach (see the survey [9]), we do not put any restrictions on the clauses.

We introduce some relations on the set of substitutions:

Definition 2.2 Let E be a set of equations, $\mathcal{X}$ be a set of variables, σ and θ be substitutions. Then σ and θ are *equal modulo E over $\mathcal{X}$*, denoted by $\sigma =_E \theta[\mathcal{X}]$, iff for every variable $x \in \mathcal{X}$ we have $E \vdash x\sigma = x\theta$. We say that σ is *more general than θ over $\mathcal{X}$*, denoted $\sigma \leq_E \theta[\mathcal{X}]$, iff there is a substitution η such that $\sigma\eta =_E \theta[\mathcal{X}]$.

The notions of a E-unifier and a complete set of E-unifiers are the main notions studied in this paper:

Definition 2.3 Let E be a set of equations, s, t be terms and $\mathcal{X}$ be a finite set of variables such that $var(s) \cup var(t) \subseteq \mathcal{X}$. A substitution θ is called an *E-unifier* of terms s and t iff $E \vdash s\theta = t\theta$. A set Θ of substitutions is a *complete set of E-unifiers for s and t away from $\mathcal{X}$* iff

1. For every $\theta \in \Theta$ we have $dom(\theta) \subseteq var(s) \cup var(t)$ and $ran(\theta) \cap \mathcal{X} = \emptyset$;
2. Every $\theta \in \Theta$ is a E-unifier of s and t;
3. For every E-unifier σ of s and t there exist $\theta \in \Theta$ such that $\theta \leq_E \sigma[\mathcal{X}]$.

Section 3

SLDE†-resolution

Let $\mathcal{P}$ be an equational Horn-clause program. Then $E_{\mathcal{P}}$ denotes the set of all equations occurring in the heads of clauses of $\mathcal{P}$. Following [6], for any finite set of equations E, we denote by $\mathcal{UNIF}_E$ a procedure that, given terms s, t and a set of variables $\mathcal{X}$ such that $var(s) \cup var(t) \subseteq \mathcal{X}$, generates a complete set $\mathcal{UNIF}_E(s, t, \mathcal{X})$ of E-unifiers for s and t away from $\mathcal{X}$. We refer the reader to [6, 7] for the discussion of such procedures.

We assume that one such procedure $\mathcal{UNIF}_E$ is fixed. The following definition of SLDE†-resolution depends on such a procedure. This definition is a specialization of the definition of [6] for the case when all atoms are equations.

Definition 3.1 Let G, G' be goals. Then G' is a *successor* of G w.r.t. $\mathcal{P}$ iff

1. The goal G has the form $\;\text{:-}\; A_1, \ldots, A_{k-1}, s = t, A_{k+1}, \ldots, A_n$;

2. There is a finite sequence

$$\mathcal{D} = (s_1 = t_1 \;\text{:-}\; \Gamma_1, \ldots, s_m = t_m \;\text{:-}\; \Gamma_m)$$

   of variants of clauses in $\mathcal{P}$ (having pairwise disjoint sets of variables and variables disjoint from the set of variables in G), a finite sequence $\theta_1, \ldots, \theta_m$ of substitutions and a substitution $\sigma \in \mathcal{UNIF}_{E_{\mathcal{P}}}(s, t, \mathcal{X})$, where $\mathcal{X}$ are all variables of $\mathcal{D}$ and G such that

$$s\sigma \Leftrightarrow_{[s_1=t_1,\theta_1]} \cdots \Leftrightarrow_{[s_m=t_m,\theta_m]} t\sigma$$

   and G' is

$$\text{:-}\; A_1\sigma, \ldots, A_{k-1}\sigma, \Gamma_1\theta_1, \ldots, \Gamma_m\theta_m, A_{k+1}\sigma, \ldots, A_n\sigma$$

Definition 3.2 An SLDE†-*derivation* for a goal G and a program $\mathcal{P}$ is any sequence of goals $G_1, \ldots, G_n$ such that $G = G_1$ and every G_{i+1} for $i \in \{1, \ldots, n-1\}$ is a successor of G_i w.r.t. $\mathcal{P}$. An SLDE†-*refutation* for G and $\mathcal{P}$ is any such derivation with $G_n = \Box$.

In [6] SLDE†-resolution has been proved complete for the so-called well-behaved programs. It has been an open problem whether SLDE†-resolution is complete for arbitrary equational Horn-clause programs. The completeness was conjectured in [4]. Here we prove that SLDE†-resolution is incomplete. We announced this result in [2]. The construction given below is more simple than the one used in [2].

Theorem 3.1 SLDE†-resolution is incomplete.

Proof. We shall use a binary function symbol $\cdot$ written in the infix notation $s \cdot t$. For any term t, by t^n we (ambiguously) denote any term of the form

$$\underbrace{t \cdot \ldots \cdot t}_{n \text{ occurrences of } t}$$

where parentheses can be put in an arbitrary order. For example, a^3 may denote both $a \cdot (a \cdot a)$ and $(a \cdot a) \cdot a$.

Consider the following program $\mathcal{P}$, consisting of a single clause

$$v \cdot v = v \text{ :- } v = f(a \cdot a)$$

and the goal $G = \text{ :- } f(x \cdot x) = f(x \cdot x) \cdot f(y)$. Then $E_\mathcal{P} = \{v \cdot v = v\}$. It is obvious that the terms $f(x \cdot x)$ and $f(x \cdot x) \cdot f(y)$ are not unifiable. They are, however, $E_\mathcal{P}$-unifiable. Applying the E-unification algorithm from [11], we obtain a complete set of $E_\mathcal{P}$-unifiers for these terms away from $\{x, y\}$ consisting of one element $\sigma = \{z/x, z/y\}$. Thus, any successor G' of G has the form

$$\text{:- } v_1\theta_1 = f(a \cdot a), \ldots, v_n\theta_n = f(a \cdot a),$$

such that

$$s_0 \Leftrightarrow_{[v_1 \cdot v_1 = v_1, \theta_1]} \cdots \Leftrightarrow_{[v_n \cdot v_n = v_n, \theta_n]} s_n$$

where $s_0 \approx f(z \cdot z)$ and $s_n \approx f(z \cdot z) \cdot f(z)$. We prove that there are $i, j \in \{1, \ldots, n\}$ such that $\theta_i = \{z/v_i\}$ and $\theta_j = \{f(z^k)/v_j\}$, for some $k \geq 1$. Define j to be the least number such that s_j contains two occurrences of the symbol f. Then obviously, $v_j\theta_j \approx f(z^k)$, for some k. Define i to be the least number such that s_i contains an occurrence of the term $f(z)$. Then obviously, $v_i\theta_i \approx z$.

Hence, G' contains two equations of the form $f(z^k) = f(a \cdot a)$ and $z = f(a \cdot a)$. It is easy to see that

$$CL(\mathcal{P}) = CL(\{(f(a \cdot a))^m = (f(a \cdot a))^n \mid m, n \geq 1\})$$

Using soundness of SLDE†-resolution, we obtain that there is no refutation for the goal G' and $\mathcal{P}$, and hence there is no refutation for $\text{ :- } f(x \cdot x) = f(x \cdot x) \cdot f(y)$.

□

It is easy to see that there is a correct answer substitution $\{a/x, a \cdot a/y\}$. To refute the goal G by SLDE†-resolution, we have to use the $E_\mathcal{P}$-unifier $\sigma' = \{z \cdot z/y, z/x\}$, equal to σ modulo E over $\{x, y\}$. With this $E_\mathcal{P}$-unifier, the goal G can be reduced to $\text{ :- } f(z \cdot z) = f(a \cdot a)$, which has a trivial refutation.

There are even more simple examples of the incompleteness of SLDE†-resolution based on the fact that the set of *all* equations $E_\mathcal{P}$ from the heads of clauses in $\mathcal{P}$ is used for generating the complete sets of E-unifiers. For example, consider the program

$$\begin{array}{l} x = y \text{ :- } c = d \\ a = b \end{array}$$

and the goal $\text{ :- } a = z$. The set $\{\varepsilon\}$ is a complete set of unifiers for a and z w.r.t. $\{x = y, a = b\}$. To prove $a = z$ using instances of $x = y$ and $a = b$, we have to use at least one instance of $x = y$. Then the goal $\text{ :- } a = z$ will be reduced to a goal containing $c = d$ which has no refutation.

Our example used in the proof of Theorem 3.1 shows that SLDE†-resolution is incomplete even when we consider a complete set of E-unifiers generated by $\mathcal{UNIF}_E$, where E is any subset of $E_\mathcal{P}$.

Section 4

Fixpoint theory

Here we consider several approaches to defining a suitable fixpoint theory for equational Horn-clause programs based on the notion of complete sets of E-unifiers. Such a fixpoint theory in the non-equational case based on the notion of a most general unifier has been developed in [3] as so-called S-semantics and C-semantics.

Instead of dealing with ground atoms, S-semantics and C-semantics deal with arbitrary atoms. This allows one to use most general unifiers in the definition of the immediate consequence operator. We introduce a definition of an equational C-interpretation similar to that of a C-interpretation of [3].

Definition 4.1 A *C-interpretation* is any set of equations of the signature Σ closed under derivability in first-order logic with equality.

In the definition of C-semantics in [3], it is required that C-interpretations be upward-closed, i.e. closed under instances. The upward-closedness is a consequence of our definition, since the formula $s\sigma = t\sigma$ is derivable from $s = t$.

For any set of clauses $\mathcal{D}$, we denote

$$CL(\mathcal{D}) \rightleftharpoons \{s = t \mid \mathcal{D} \vdash s = t\}$$

It means that the quasivariety satisfying $\mathcal{D}$ and the variety satisfying $CL(\mathcal{D})$ have the same freely generated systems [16].

In order to define the family of the immediate consequence operators, we change the procedure $\mathcal{UNIF}_E$ in the following way. First, we omit the third argument assuming that $\mathcal{UNIF}_E$ returns substitutions away from all relevant variables. Second, $\mathcal{UNIF}_E$ will be applied to a set of pairs of terms $\{\langle s_1, t_1\rangle, \ldots, \langle s_n, t_n\rangle\}$ instead of a pair of terms and return a complete set of simultaneous E-unifiers for this set.

A natural generalization of the immediate consequence operator for C-semantics based on the complete sets of unifiers is the following:

Definition 4.2 The *immediate consequence operator* $T^{\dagger}_{\mathcal{P}}$ is defined as follows. For any C-interpretation $\mathcal{I}$ we have

$$\begin{array}{ll} T^{\dagger}_{\mathcal{P}}(\mathcal{I}) \rightleftharpoons CL(\{s\sigma = t\sigma \mid & \text{there are } s_1, \ldots, s_n, t_1, \ldots, t_n \\ & \text{such that } (s = t \;\texttt{:-}\; s_1 = t_1, \ldots, s_n = t_n) \in \mathcal{P} \\ & \text{and } \sigma \in \mathcal{UNIF}_{\mathcal{I}}(\{\langle s_1, t_1\rangle, \ldots, \langle s_n, t_n\rangle\})\}) \end{array}$$

Proposition 4.1 The operator $T^{\dagger}_{\mathcal{P}}$ is not monotone.

Proof. Let the signature Σ contain a binary function symbol $\cdot$ and two constants a, b. Consider two interpretations $\mathcal{I}_1$ and $\mathcal{I}_2$ defined by

$$\begin{array}{l} \mathcal{I}_1 \rightleftharpoons CL(\{(x \cdot y) \cdot z = x \cdot (y \cdot z)\}) \\ \mathcal{I}_2 \rightleftharpoons CL(\{(x \cdot y) \cdot z = x \cdot (y \cdot z), x \cdot x = x\}) \end{array}$$

Evidently, $\mathcal{I}_1 \subseteq \mathcal{I}_2$. Consider the program $\mathcal{P}$ consisting of one clause

$$x = b \text{ :- } x \cdot a = a \cdot x$$

Then

$$\begin{array}{l} \mathcal{UNIF}_{\mathcal{I}_1}(\{\langle x \cdot a, a \cdot x \rangle\}) = \{\{a^n/x\} \mid n \geq 1\} \\ \mathcal{UNIF}_{\mathcal{I}_2}(\{\langle x \cdot a, a \cdot x \rangle\}) = \{\{a/x\}\} \end{array}$$

Thus,

$$\begin{array}{l} T^{\dagger}_{\mathcal{P}}(\mathcal{I}_1) = CL(\{a^n = b \mid n \geq 1\}) \\ T^{\dagger}_{\mathcal{P}}(\mathcal{I}_2) = CL(\{a = b\}) \end{array}$$

We have $T^{\dagger}_{\mathcal{P}}(\mathcal{I}_1) \not\subseteq T^{\dagger}_{\mathcal{P}}(\mathcal{I}_2)$.

□

Besides non-monotonicity, this natural operator has other unpleasant properties. For example, the least C-model of a program $\mathcal{P}$, i.e. $CL(\mathcal{P})$, may be not a fixpoint of this operator. Consider the program of Theorem 3.1 consisting of one clause $v \cdot v = v$:- $v = f(a \cdot a)$. Then

$$CL(\mathcal{P}) = CL(\{(f(a \cdot a))^m = (f(a \cdot a))^n \mid m, n \geq 1\})$$

Denote $CL(\mathcal{P})$ by $\mathcal{I}$. The set consisting of one substitution $\{(f(a \cdot a))^2/v\}$ is a complete set of $\mathcal{I}$-unifiers for $v = f(a \cdot a)$. We can assume that this complete set is generated by a procedure $\mathcal{UNIF}_{\mathcal{I}}$ so that $\mathcal{UNIF}_{\mathcal{I}}(\{\langle v, f(a \cdot a)\rangle\}) = \{\{(f(a \cdot a))^2/v\}\}$. By the definition of $T^{\dagger}_{\mathcal{P}}$ we have

$$\begin{array}{rl} T^{\dagger}_{\mathcal{P}}(\mathcal{I}) & = CL(\{(f(a \cdot a))^4 = (f(a \cdot a))^2\}) \\ & = CL(\{(f(a \cdot a))^{2m} = (f(a \cdot a))^{2n} \mid m, n \geq 1\}) \end{array}$$

Hence, $T^{\dagger}_{\mathcal{P}}(\mathcal{I}) \neq \mathcal{I}$.

Now we shall introduce an immediate consequence operator having all desirable properties. To this end, we consider the set of all E-unifiers, as in the definition of SLDE-resolution.

$$\begin{array}{ll} T_{\mathcal{P}}(\mathcal{I}) \rightleftharpoons CL(\{s\sigma = t\sigma \mid & \text{there are } s_1, \ldots, s_n, t_1, \ldots, t_n \\ & \text{such that } (s = t \text{ :- } s_1 = t_1, \ldots, s_n = t_n) \in \mathcal{P} \\ & \text{and for all } i \in \{1, \ldots, n\} \text{ we have } (s_i\sigma = t_i\sigma) \in \mathcal{I}\}) \end{array}$$

Theorem 4.1 The operator $T_{\mathcal{P}}$ is monotone and ω-continuous. There is the least fixpoint $\mathcal{I}$ of $T_{\mathcal{P}}$ such that $\mathcal{I} = \bigcup_{i=0}^{\omega} \mathcal{I}_i$, where

$$\begin{array}{l} \mathcal{I}_0 \rightleftharpoons \{t = t \mid t \in \mathcal{T}_{\Sigma}\} \\ \mathcal{I}_{i+1} \rightleftharpoons T_{\mathcal{P}}(\mathcal{I}_i) \end{array}$$

In addition, $(s = t) \in \mathcal{I}$ iff $\mathcal{P} \vdash s = t$.

Proof. Monotonicity is obvious. Let us prove the ω-continuity of $T_{\mathcal{P}}$. Let $\mathcal{J}_0 \subseteq \mathcal{J}_1 \subseteq \ldots$ be a family of C-interpretations. We have to prove that $T_{\mathcal{P}}(\bigcup_{i\in\omega} \mathcal{J}_i) = \bigcup_{i\in\omega} T_{\mathcal{P}}(\mathcal{J}_i)$. The inclusion $T_{\mathcal{P}}(\bigcup_{i\in\omega} \mathcal{J}_i) \supseteq \bigcup_{i\in\omega} T_{\mathcal{P}}(\mathcal{J}_i)$ is straightforward by monotonicity.

We prove the inclusion $T_{\mathcal{P}}(\bigcup_{i\in\omega} \mathcal{J}_i) \subseteq \bigcup_{i\in\omega} T_{\mathcal{P}}(\mathcal{J}_i)$. Denote $\bigcup_{i\in\omega} \mathcal{J}_i$ by $\mathcal{J}$. Suppose $(s = t) \in T_{\mathcal{P}}(\mathcal{J})$. By the definition of $T_{\mathcal{P}}$ and by compactness, there are clauses in $\mathcal{P}$

$$\begin{array}{c} s_1 = t_1 \ \texttt{:-}\ s_1^1 = t_1^1, \ldots, s_1^{k_1} = t_1^{k_1} \\ \vdots \\ s_n = t_n \ \texttt{:-}\ s_n^1 = t_n^1, \ldots, s_n^{k_n} = t_n^{k_n} \end{array}$$

and substitutions $\sigma_1, \ldots, \sigma_n$ such that $\{s_1\sigma_1 = t_1\sigma_1, \ldots, s_n\sigma_n = t_n\sigma_n\} \vdash s = t$ and $\mathcal{J} \vdash s_i^j\sigma_i = t_i^j\sigma_i$, for all $i \in \{1, \ldots, j\}$ and $j \in \{1, \ldots, k_i\}$. By compactness, there are interpretations $\mathcal{J}_{i_1}, \ldots, \mathcal{J}_{i_l}$ such that $\mathcal{J}_{i_1} \cup \ldots \cup \mathcal{J}_{i_l} \vdash s_i^j\sigma_i = t_i^j\sigma_i$, for all $i \in \{1, \ldots, j\}$ and $j \in \{1, \ldots, k_i\}$. Define $m \rightleftharpoons \max(i_1, \ldots, i_l)$. Then $(s = t) \in T_{\mathcal{P}}(\mathcal{J}_m) \subseteq \bigcup_{i\in\omega} T_{\mathcal{P}}(\mathcal{J}_i)$.

The rest of the claim is proved by standard arguments using the ω-continuity.

□

For technical reasons, we define here one more operator not based on the complete sets of E-unifiers. This operator was introduced in [13]. Let, for any C-interpretation $\mathcal{I}$,

$$\begin{array}{lll} K_{\mathcal{P}}(\mathcal{I}) \rightleftharpoons CL(\mathcal{I} \cup \{s\sigma = t\sigma \mid & \text{there are } s_1, \ldots, s_n, t_1, \ldots, t_n \\ & \text{such that } (s = t \ \texttt{:-}\ s_1 = t_1, \ldots, s_n = t_n) \in \mathcal{P} \\ & \text{and for all } i \in \{1, \ldots, n\} \text{ we have } (s_i\sigma = t_i\sigma) \in \mathcal{I}\}) \end{array}$$

The following result was claimed in [13]:

Theorem 4.2 The operator $K_{\mathcal{P}}$ is monotone and ω-continuous. There is the least fixpoint $\mathcal{I}$ of $K_{\mathcal{P}}$ such that $\mathcal{I} = \bigcup_{i=0}^{\omega} \mathcal{I}_i$, where

$$\begin{array}{l} \mathcal{I}_0 \rightleftharpoons \{t = t \mid t \in \mathcal{T}_\Sigma\} \\ \mathcal{I}_{i+1} \rightleftharpoons K_{\mathcal{P}}(\mathcal{I}_i) \end{array}$$

In addition, $(s = t) \in \mathcal{I}$ iff $\mathcal{P} \vdash s = t$.

Proof. Similar to that of Theorem 4.1. □

We introduce another operator based on the complete sets of E-unifiers.

Definition 4.3 The *immediate consequence operator* $T_{\mathcal{P}}^{\dagger\dagger}$ is defined as follows. For any C-interpretation $\mathcal{I}$ we have

$$\begin{array}{lll} T_{\mathcal{P}}^{\dagger\dagger}(\mathcal{I}) \rightleftharpoons CL(\mathcal{I} \cup \{s\sigma = t\sigma \mid & \text{there are } s_1, \ldots, s_n, t_1, \ldots, t_n \\ & \text{such that } (s = t \ \texttt{:-}\ s_1 = t_1, \ldots, s_n = t_n) \in \mathcal{P} \\ & \text{and } \sigma \in \mathcal{UNIF}_{\mathcal{I}}(\{\langle s_1, t_1\rangle, \ldots, \langle s_n, t_n\rangle\})\}) \end{array}$$

This operator has all standard properties:

Theorem 4.3 The operator $T_{\mathcal{P}}^{\dagger\dagger}$ is monotone and ω-continuous. There is the least fixpoint $\mathcal{I}$ of $T_{\mathcal{P}}^{\dagger\dagger}$ such that $\mathcal{I} = \bigcup_{i=0}^{\omega} \mathcal{I}_i$, where

$$\begin{array}{l} \mathcal{I}_0 \rightleftharpoons \{t = t \mid t \in \mathcal{T}_\Sigma\} \\ \mathcal{I}_{i+1} \rightleftharpoons T_{\mathcal{P}}^{\dagger\dagger}(\mathcal{I}_i) \end{array}$$

In addition, $(s = t) \in \mathcal{I}$ iff $\mathcal{P} \vdash s = t$.

Proof. We prove that for every C-interpretation $\mathcal{I}$ we have $T^{\dagger\dagger}_{\mathcal{P}}(\mathcal{I}) = K_{\mathcal{P}}(\mathcal{I})$ and then apply Theorem 4.2.

It is evident that $T^{\dagger\dagger}_{\mathcal{P}}(\mathcal{I}) \subseteq K_{\mathcal{P}}(\mathcal{I})$. Let us prove $K_{\mathcal{P}}(\mathcal{I}) \subseteq T^{\dagger\dagger}_{\mathcal{P}}(\mathcal{I})$. Let $(s = t) \in K_{\mathcal{P}}(\mathcal{I})$. By compactness, there is a finite number of substitutions $\sigma_1, \ldots, \sigma_n$ and a finite number of clauses in $\mathcal{P}$

$$\begin{array}{c} s_1 = t_1 \text{ :- } s_1^1 = t_1^1, \ldots, s_1^{k_1} = t_1^{k_1} \\ \vdots \\ s_n = t_n \text{ :- } s_n^1 = t_n^1, \ldots, s_n^{k_n} = t_n^{k_n} \end{array}$$

such that $\mathcal{I} \cup \{s_1\sigma_1 = t_1\sigma_1, \ldots, s_n\sigma_n = t_n\sigma_n\} \vdash s = t$ and $\mathcal{I} \vdash s_i^j\sigma_i = t_i^j\sigma_i$, for all $i \in \{1, \ldots, j\}$ and $j \in \{1, \ldots, k_i\}$. By the definition of a complete set of unifiers, there are substitutions $\theta_1, \ldots, \theta_n$ such that for all $i \in \{1, \ldots, n\}$ we have

1. $\theta_i \in \mathcal{UNIF}_{\mathcal{I}}(\{\langle s_i^1, t_i^1\rangle, \ldots, \langle s_i^{k_i}, t_i^{k_i}\rangle\})$;
2. $\theta_i \leq_{\mathcal{I}} \sigma_i$.

From 2 it follows that for all $i \in \{1, \ldots, n\}$ we have $\mathcal{I}, s_i\theta_i = t_i\theta_i \vdash s_i\sigma_i = t_i\sigma_i$. Hence, $\mathcal{I} \cup \{s_1\theta_1 = t_1\theta_1, \ldots, s_n\theta_n = t_n\theta_n\} \vdash s = t$. By the definition of $T^{\dagger\dagger}_{\mathcal{P}}$, we have $(s_i\theta_i = t_i\theta_i) \in T^{\dagger\dagger}_{\mathcal{P}}(\mathcal{I})$. Hence, $\mathcal{I} \cup T^{\dagger\dagger}_{\mathcal{P}}(\mathcal{I}) \vdash s = t$. From $\mathcal{I} \subseteq T^{\dagger\dagger}_{\mathcal{P}}(\mathcal{I})$ we get $T^{\dagger\dagger}_{\mathcal{P}}(\mathcal{I}) \vdash s = t$. Since $T^{\dagger\dagger}_{\mathcal{P}}(\mathcal{I})$ is closed under derivability, we have $(s = t) \in T^{\dagger\dagger}_{\mathcal{P}}(\mathcal{I})$.

$\square$

Bibliography

[1] G. Birkhoff. Subdirect unions in universal algebras. *Bull. Amer. Math. Soc.*, 50:764–768, 1944.

[2] A. Degtyarev and A. Voronkov. A new procedural interpretation of Horn clauses with equality. In Leon Sterling, editor, *Proceedings of the Twelfth International Conference on Logic Programming*, pages 565–579. The MIT Press, 1995.

[3] M. Falaschi, G. Levi, M. Martelli, and C. Palamidessi. Declarative modeling of the operational behavior of logic languages. *Theoretical Computer Science*, 69(3):289–318, 1989.

[4] U. Furbach, S. Hölldobler, and J. Schreiber. Horn equality theories and paramodulation. *Journal of Automated Reasoning*, 5:309–337, 1989.

[5] J.H. Gallier and S. Raatz. Extending SLD-resolution methods for Horn clauses with equality based on E-unification. In *Symposium on Logic Programming*, pages 168–179, 1986.

[6] J.H. Gallier and S. Raatz. Extending SLD-resolution to equational Horn clauses using E-unification. *Journal of Logic Programming*, 6(3):3–44, 1989.

[7] J.H. Gallier and W. Snyder. Complete sets of transformations for general E-unification. *Theoretical Computer Science*, 67:203–260, 1989.

[8] J.A. Goguen and J. Meseguer. EQLOG: Equality, types and generic modules for logic programming. In D. DeGroot and G. Lindstrom, editors, *Logic Programming, Functions, Relations and Equations*, pages 295–363. Prentice Hall, 1986.

[9] M. Hanus. The integration of functions into logic programming: from theory to practice. *Journal of Logic Programming*, 19,20:583–628, 1994.

[10] S. Hölldobler. *Foundations of Equational Logic Programming*, volume 353 of *Lecture Notes in Artificial Intelligence*. Springer Verlag, 1989.

[11] G. Huet and D.C. Oppen. Equations and rewrite rules. In Book, editor, *Formal Languages: Perspectives and Open Problems*. Academic Press, 1980.

[12] J. Jaffar, J.-L. Lassez, and M.J. Maher. A theory of complete logic programs with equality. *Journal of Logic Programming*, 1(3):211–223, 1983.

[13] S. Kaplan. Conditional rewrite rules. *Theoretical Computer Science*, 33:175–193, 1984.

[14] J.W. Lloyd. *Foundations of Logic Programming (2nd edition)*. Springer Verlag, 1987.

[15] A. Maĺcev. Subdirect products of models. *Dokl. Akad. Nauk SSSR (in Russian)*, 109:264–266, 1956.

[16] A. Maĺcev. *Algebraic Systems.* Springer Verlag, 1973.

[17] G. Plotkin. Building-in equational theories. In Meltzer and Michie, editors, *Machine Intelligence*, volume 7, pages 73–90. Edinburgh University Press, Edinburgh, 1972.

[18] J.C. Sheperdson. SLDNF-resolution with equality. *Journal of Automated Reasoning*, 8:297–306, 1992.

[19] M. Thielscher. On the completeness of SLDENF-resolution. *Journal of Automated Reasoning.* To appear.