I N F O R M A T I K

Unification in Extensions of Shallow Equational Theories

Florent Jacquemard
Christoph Meyer
Christoph Weidenbach

MPI–I–98–2–002 January 1998

FORSCHUNGSBERICHT RESEARCH REPORT

M A X - P L A N C K - I N S T I T U T
F Ü R
I N F O R M A T I K

Im Stadtwald 66123 Saarbrücken Germany

Authors' Addresses

Florent Jacquemard
LORIA and INRIA
615 rue du Jardin Botanique
B.P. 101, 54602 Villers-les-Nancy Cedex, France
Florent.Jacquemard@loria.fr

Christoph Meyer
Max-Planck-Institut für Informatik
Im Stadtwald
66123 Saarbrücken
meyer@mpi-sb.mpg.de

Christoph Weidenbach
Max-Planck-Institut für Informatik
Im Stadtwald
66123 Saarbrücken
weidenb@mpi-sb.mpg.de

Publication Notes

The present report has been submitted for publication elsewhere and will be copyrighted if accepted.

Acknowledgements

We have to thank our reviewers at RTA-98 whose constructive criticism lead to significant improvements of the conference paper version of this report and finally this report itself. Thanks to Harald Ganzinger for many valuable discussions and to Jürgen Stuber for comments and suggestions on this work.

Abstract

We show that unification in certain extensions of shallow equational theories is decidable. Our extensions generalize the known classes of shallow or standard equational theories. In order to prove decidability of unification in the extensions, a class of Horn clause sets called sorted shallow equational theories is introduced. This class is a natural extension of tree automata with equality constraints between brother subterms as well as shallow sort theories. We show that saturation under sorted superposition is effective on sorted shallow equational theories. So called semi-linear equational theories can be effectively transformed into equivalent sorted shallow equational theories and generalize the classes of shallow and standard equational theories.

Keywords

E-Unification, Tree automata, Monadic Theories, Superposition, Decidability

1 Introduction

Algorithms to solve unification and word problems in an equational theory play a crucial role in many areas of computer science like automated deduction, logic and functional programming, and symbolic constraint solving. Many algorithms are dedicated to particular theories and often semantic conditions are assumed. In addition, a lot of progress has been made towards syntactic characterizations of classes of equational theories or rewrite systems in which these problems are decidable. This is the case of unification in ground theories (Kozen 1981) or of the word problem with respect to left-linear right-ground rewrite systems (Oyamaguchi 1990). The class of *shallow* theories, axiomatized by equations in which variables occur at most at depth one, has been shown by Comon, Haberstrau & Jouannaud (1994) to have a decidable unification problem. They exploit a transformation of the system into an equivalent cycle-syntactic presentation (Kirchner 1986). By a termination analyses under basic superposition Nieuwenhuis (1996) generalized the result to so-called standard theories.

Furthermore, tree automata and tree grammars have also been used for unification purposes. Limet & Réty (1997) use Tree Tuple Synchronized Grammars to generate solutions to unification problems by a simulation of narrowing. In (Kaji, Toru & Kasami 1997) it is shown that the closure with respect to some kind of term rewriting system of the (recognizable set) of ground instances of a linear term is recognizable. Similar techniques based on the completion of tree automata are presented by Comon (1995) and Jacquemard (1996) for linear shallow TRS and a generalization called linear growing TRS. The decidability of the word problem as well as restricted cases of unifiability in the concerned theories can be derived from these results.

In this paper we show the decidability of unification in so-called semi-linear equational theories which strictly extend shallow theories. Informally, a semi-linear system contains equations in which non-linear variables only appear in the same subterms. For example, the equation $f(f(x,x),y) \approx g(f(x,x))$ is semi-linear whereas $f(g(x),h(x),h(g(y))) \approx h(g(x))$ is not. Our techniques are influenced by tree automata, sorted unification and saturation-based methods.

Sorted shallow equational theories naturally generalize tree automata with equality constraints (Bogaert & Tison 1992) as well as shallow sort theories (Weidenbach 1998). Throughout the paper, we consider the following example of Nieuwenhuis (1996). The equational theory is given by the equations $f(g(x),y) \approx h(y)$ and $f(x,x) \approx g(x)$. Nieuwenhuis' definition of standard theories does not include this case. The closure of the theory under basic superposition, the calculus he suggests, leads to an infinite set of equations $g(h^n(g(x))) \approx h^{n+1}(g(x))$. The infinite expansion can be avoided by abstracting the linear (semi-linear) term $g(x)$ into a sort declaration $S(g(x))$. The theory is then transformed into a sorted shallow equational theory consisting of the Horn clauses $\| \to S(g(x))$, $S(x) \| \to f(x,y) = h(y)$, $\| \to f(x,x) = g(x)$. Our notation for clauses is of the form *Sort Constraint* $\|$ *Antecedent* $\to$ *Succedent* where the sort constraint atoms are particular, monadic antecedent atoms for which specific inference rules are provided by our sorted superposition calculus.

The paper is organized as follows: Section 3 starts with a discussion on tree automata with brother constraints. We prove that they are not sufficient for our purpose. Then sorted shallow equational theories are studied. It is shown that saturation under sorted superposition terminates and that unifiability modulo the saturated theory is decidable. A procedure which transforms a sorted semi-linear equational theory into an equivalent sorted shallow one is given in Section 4. This implies the decidability of unifiability modulo a set of (sorted) semi-linear equations. This result strictly embeds previous ones concerning shallow theories by Comon et al. (1994). We show in Section 5 that with similar techniques, we can treat a generalization of standard theories as proposed by Nieuwenhuis (1996). In the same section, we also show that our decidability results are close to the border between decidability/undecidability and discuss some related work on E-unification.

2 Preliminaries

We adhere to the usual definitions for variables, terms, substitutions, equations, atoms, (positive and negative) literals, multisets, and clauses, see Dershowitz & Jouannaud (1990) for what concerns equational theories. We give just the most important definitions for our purpose.

The algebra of terms over a finite set of function symbols $\mathcal{F}$ and a set $\mathcal{X}$ of variables is denoted $\mathcal{T}(\mathcal{F},\mathcal{X})$ and $\mathcal{T}(\mathcal{F})$ is its subalgebra of ground terms. An *equation* is an unoriented pair of terms of $\mathcal{T}(\mathcal{F},\mathcal{X})$ denoted $s \approx t$. For sake of simplicity, we may apply to equations or other atoms the same following notations as for terms. The function *vars* maps terms, atoms, literals, clauses and sets of such objects to the set of variables occurring in these objects. A *position* p in a term (equation, atom) is a word over the natural numbers. For a term (equation, atom) t we define $t|_p$ of t at position p by $t|_\epsilon = t$ and $t|_{i.p} = t_i|_p$ where $t = f(t_1,\ldots,t_n)$ and $1 \leq i \leq n$. We write $t[s]_p$ to denote that $t|_p = s$ and $t[p/s']$ is the term obtained from t by replacing its subterm at position p by s'.

A term is called *complex* if it is neither a constant nor a variable. A term t is called *shallow* if t is a variable or is of the form $f(x_1,\ldots,x_n)$ where the x_i are not necessarily distinct. An equation $s \approx t$ is called *shallow* if both s and t are shallow. Note that shallow variables in $s \approx t$ can be arbitrarily shared by s and t. A term t is called *linear* if every variable occurs at most once in t. A term t is called *semi-linear* if it is a variable or of the form $f(t_1,\ldots,t_n)$

such that every t_i is semi-linear and whenever $vars(t_i) \cap vars(t_j) \neq \emptyset$ we have $t_i = t_j$ for all i, j. An equation $s \approx t$ is semi-linear if (i) s and t are variables or (ii) $s = f(s_1, \ldots, s_n)$ and (a) if t is a variable and $t \in vars(s_i)$ then $s_i = t$ for all i or (b) if $t = g(t_1, \ldots, t_m)$ and $vars(s_i) \cap vars(t_j) \neq \emptyset$ then $s_i = t_j$ for all i, j. For instance, the term $f(g(x), g(x), h(y, y))$ and the equations $h(g(x), g(x), y) \approx f(y, g(x), y)$ and $f(g(x), g(x), y) \approx y$ are semi-linear, but $f(g(x), g(x), h(x, y))$, $h(g(x), x)$, and $h(g(x), g(x)) \approx x$ are not.

Atoms formed from unary predicates are called *monadic*. For the purpose of this paper, a clause is written in the form $\Theta \parallel \Gamma \rightarrow \Delta[\![P]\!]$ where the *sort constraint* Θ is a multiset of monadic atoms representing the sort restrictions, the *term constraint* P is a conjunction of syntactic equations of the form $s = t$ and the multisets Γ and Δ denote the antecedent and succedent atoms of the clause, respectively. If P is empty we simply omit the term constraint. Semantically, a clause $\Theta \parallel \Gamma \rightarrow \Delta[\![P]\!]$ is interpreted as $\Theta\sigma \parallel \Gamma\sigma \rightarrow \Delta\sigma$ where σ is the syntactic most general unifier of P, denoted by $mgu(P)$. As usual, all variables are assumed to be universally quantified and the clause is interpreted as an implication where the conjunction of all sort constraint and antecedent atoms implies the disjunction of all succedent atoms. If there exists no mgu σ for P, the clause is a tautology and can therefore always be removed in our context. We say that the constraint P is in *solved form*, if P is of the form $x_1 = t_1 \wedge \ldots \wedge x_n = t_n$ where $x_i \neq x_j$ for all i, j and $x_i \notin vars(t_k)$ for all $i \leq k \leq n$. Using the rules of syntactic unification (see, e.g., Jouannaud & Kirchner (1991)) any constraint can be transformed in polynomial time into solved form or $\perp$ indicating that the term constraint has no solution and therefore the clause can be removed. The following operations on the constraint do also not change the semantics of a clause $\Theta \parallel \Gamma \rightarrow \Delta[\![x = t \wedge P]\!]$: We can *propagate* the equation $x = t$ into Θ, Γ, Δ, and P by instantiating arbitrary occurrences of x with t. If $x \notin vars(t) \cup vars(P) \cup vars(\Theta \cup \Gamma \cup \Delta)$ then $x = t$ can be removed from P and we call the equation $x = t$ *redundant*. For the purpose of this paper, we assume that constraints are always in solved form, redundant equations are always removed from term constraints and clauses with an unsolvable term constraint are always deleted.

A partition $C_1, \ldots, C_n$ of a clause C is called a *variable component partition*, if for every pair of literals $L \in C_i$, $K \in C_j$ $(i \neq j)$ we have $vars(L) \cap vars(K) = \emptyset$ and no C_i can be further partitioned such that this condition is still satisfied. Every C_i is then called a *variable component* of the clause C.

For any initial clause set we assume that the arguments of all sort constraint atoms are variables and that all term constraints are empty. This is a necessary prerequisite for our calculus to be complete, since term positions in sort constraint atoms are always subject to the basic restriction (Weidenbach 1996). Furthermore, for the theories which we consider here it is always the case that either the antecedent or the succedent of a clause is empty.

We call a sort constraint Θ *solved* in a clause $\Theta \,\|\, \Gamma \rightarrow \Delta[\![P]\!]$ with $\sigma = mgu(P)$ if $vars(\Theta\sigma) \subseteq vars(\Gamma\sigma \cup \Delta\sigma)$ and all terms occurring in $\Theta\sigma$ are variables. A clause $T_1(x_1), \ldots, T_n(x_n) \,\| \rightarrow S(t)$ is called a *declaration* if $T_1(x_1), \ldots, T_n(x_n)$ is solved. In case t is a variable, a declaration is called a *subsort* declaration. A declaration $T_1(x_1), \ldots, T_n(x_n) \,\| \rightarrow S(t)$ is *shallow* (*linear*, *semi-linear*) if t is shallow (linear, semi-linear). A *sort theory* is a finite set of declarations. It is called *shallow* (*linear*, *semi-linear*) if all declarations are shallow (linear, semi-linear). A *sorted equation* (*sorted disequation*) is a clause $\Theta \,\| \rightarrow l \approx r$ (a clause $\Theta \,\|\, l \approx r \rightarrow$) where Θ is solved. A *sorted equational theory* is a finite set of sorted equations and declarations. It is called *shallow* (*semi-linear*) if all equations and all declarations are shallow (semi-linear).

A *substitution* is a mapping from $\mathcal{X}$ to $\mathcal{T}(\mathcal{F}, \mathcal{X})$. As usual, we do not distinguish between a substitution and its homomorphic extension in the free algebra $\mathcal{T}(\mathcal{F}, \mathcal{X})$. Given an *equational theory* E, i.e., a finite set of equations, we write $s \xrightarrow[E]{} t$ iff there exists an equation $l \approx r \in E$ and a substitution σ such that $s|_p = l\sigma$ and $t = s[p/r\sigma]$. The symmetric closure of $\rightarrow$ is denoted by $\xleftrightarrow[E]{}$ and the reflexive, symmetric and transitive closure by $\xleftrightarrow[E]{*}$. Two terms s and t are called *unifiable* modulo an equational theory E iff there exists a substitution σ such that $s\sigma \xleftrightarrow[E]{*} t\sigma$. Note that this is equivalent to stating that the clause set consisting of the equational theory E and the clause $\|\, s \approx t \rightarrow$ is unsatisfiable.

We say that a clause $\Theta \,\|\, \Gamma \rightarrow \Delta[\![P]\!]$ *subsumes* a clause $\Lambda \,\|\, \Psi \rightarrow \Pi[\![Q]\!]$, if there exists a substitution σ such that $\Theta\sigma \subseteq \Lambda\sigma$, $\Gamma\sigma \subseteq \Psi\sigma$, $\Delta\sigma \subseteq \Pi\sigma$ and $P\sigma \subseteq Q\sigma$, where we consider the term constraint to be a set of equations and we assume that matching with respect to the commutativity of $=$ and σ maps only variables to variables. Accordingly, a clause C is a condensation of some clause D, if C is a factor of D and C subsumes D. Our notion of subsumption and condensation is weaker than usual, but it is obviously compatible with the redundancy concept of basic paramodulation (Bachmair, Ganzinger, Lynch & Snyder 1995, Nieuwenhuis & Rubio 1995), the calculus we will use to develop the decidability results in Section 3.3 and Section 5.

For a set of Horn clauses $\mathcal{A}$ and a clause C, $\mathcal{A} \models C$ denotes the usual semantic entailment relation where all variables of $\mathcal{A}$ and C are assumed to be universally quantified.

3 Shallow Sorted Equational Theories

In this section we show that there exist non-linear shallow equational theories whose unification problem can be decided by saturation-based methods, but not by tree automata (with constraints) techniques.

3.1 Tree automata and linear shallow theories

We adopt here a definition of tree automata by means of Horn clauses. This definition, though non-standard, is equivalent to the usual ones, e.g., (Bogaert & Tison 1992). A systematic correspondence between various types of Horn clause sets and known classes of tree automata with constraints has been studied by Weidenbach (1998).

Definition 3.1
A tree automaton $\mathcal{A}$ is a finite set of linear shallow declarations of the form $S_1(x_1), \ldots, S_n(x_n) \parallel \rightarrow S(f(x_1, \ldots, x_n))$.

Following tree automata terminology, the unary predicates are called *states* and the Horn clauses of $\mathcal{A}$ are *transition rules* or just transitions.

Note that in Definition 3.1 not all linear shallow declarations are valid tree automata transitions.

However, Weidenbach (1998) indicates that a set containing declarations like $S_1(x_1) \parallel \rightarrow S(f(g(a), x_1))$ or subsort declarations like $S_1(x), S_2(x) \parallel \rightarrow S(x)$ can be transformed into an equivalent set of declarations like in Definition 3.1.

A term $t \in \mathcal{T}(\mathcal{F})$ is *recognized* by $\mathcal{A}$ in some state S if $\mathcal{A} \models S(t)$. If we fix in $\mathcal{A}$ a subset $\mathcal{S}$ of *final states* (final predicates), then $t \in \mathcal{T}(\mathcal{F})$ is recognized by $\mathcal{A}$ (with respect to $\mathcal{S}$) if t is recognized by $\mathcal{A}$ in some final state. A set $L \subseteq \mathcal{T}(\mathcal{F})$ is a recognizable language if L is the set of ground terms which are recognized by a tree automaton $\mathcal{A}$ (with respect to some set of final states).

The class of recognizable languages is closed under Boolean operations. Every recognizable language is recognized by some *deterministic* tree automaton $\mathcal{A}$ such that a ground term cannot be recognized by $\mathcal{A}$ in more than one state. Every recognizable language is recognized by some *completely specified* tree automaton $\mathcal{A}$, such that every ground term is recognized by $\mathcal{A}$ at least in one state. It is decidable in polynomial time whether a given term $t \in \mathcal{T}(\mathcal{F})$ is recognized by a tree automaton $\mathcal{A}$. It is decidable in linear time whether the language recognized by some tree automaton $\mathcal{A}$ is empty or not.

Tree automata and grammars have been used by Kaji et al. (1997) and Limet & Réty (1997) to solve word and unifiability problems. In the first paper as well as in the papers by Comon (1995) and Jacquemard (1996) the recognizability of the closure of some recognizable set L with respect to term rewriting systems of restricted classes is investigated. In the following we denote the closure of a set of terms $L \subseteq \mathcal{T}(\mathcal{F})$ with respect to an equational system E by $(\overset{*}{\underset{E}{\leftrightarrow}})(L) := \{s \in \mathcal{T}(\mathcal{F}) \mid \exists t \in L \; t \overset{*}{\underset{E}{\leftrightarrow}} s\}$. For a given system E we can reduce the word problem $s \overset{*}{\underset{E}{\leftrightarrow}} t$ to the membership problem for $s \in (\overset{*}{\underset{E}{\leftrightarrow}})(\{t\})$ if the closure set and L is recognizable. For a goal $s = t$ where s and t are both linear and $vars(s) \cap vars(t) = \emptyset$, unifiability modulo

E is equivalent to $\{s\sigma \mid \sigma \text{ ground}\} \cap (\xleftrightarrow[E]{*})(\{t\sigma \mid \sigma \text{ ground}\}) \neq \emptyset$. Since the set of ground instances of s and t are both recognizable, unifiability in this case can be reduced to an emptiness decision problem for tree automata.

We call an equational system E a system *with ground* terms, if arbitrary ground terms are allowed to occur in E. For many equational systems, like sorted shallow systems, the extension to ground terms does not increase expressivity, since any ground term can be generated by a finite number of linear shallow declarations.

Theorem 3.2 ((Comon 1995))
Let E be a linear shallow equational system with ground terms and L be a recognizable language. Then $(\xleftrightarrow[E]{*})(L)$ is a recognizable language.

The principle of the construction for linear shallow equational systems is the following. We start with a tree automaton $\mathcal{A}_0$ which recognizes L and contains one state S_{l_i} for each direct ground subterm l_i in equations $f(l_1, \ldots, l_n) \approx r$ in E such that l_i (and only l_i) is recognized by $\mathcal{A}_0$ in S_{l_i}. In some sense these subterms are abstracted by $\mathcal{A}_0$. Then $\mathcal{A}_0$ is completed with respect to inference rules like the following:

$$\text{Inf}\ \frac{\begin{array}{r}\| \rightarrow f(l_1, \ldots, l_n) \approx g(r_1, \ldots, r_m) \\ S_1(x_1), \ldots, S_n(x_n) \,\| \rightarrow S(f(x_1, \ldots, x_n))\end{array}}{T_1(x_1), \ldots, T_m(x_m) \,\| \rightarrow S(g(x_1, \ldots, x_m))}$$

where $f(l_1, \ldots, l_n) \approx g(r_1, \ldots, r_m) \in E$ is a linear shallow equation with ground terms, if l_j is a ground term then it is recognized by S_j and the T_i are constructed in the following way: If r_i is a ground term, then $T_i = S_{r_i}$ and if r_i is a variable that is equal to some variable l_j, then $T_i = S_j$. If we apply paramodulation to the premises of the above inference rule we obtain the clause $S_1(l_1), \ldots, S_n(l_n) \,\| \rightarrow S(g(r_1, \ldots, r_n))$. With the above conditions, this clause is equivalent to $T_1(y_1), \ldots, T_m(y_m) \,\| \rightarrow S(g(y_1, \ldots, y_m))$. This relates the automata theoretic approach and its generalization presented in Section 3.3. Unfortunately, the above recognizability result of Theorem 3.2 cannot be extended to non-linear systems.

Lemma 3.3
There exists a recognizable set L and a (non-linear) shallow equational system E such that $(\xleftrightarrow[E]{*})(L)$ is not recognizable.

Proof. Assume f is a binary function symbol, s is unary and a is a constant, and let $L = \{a\}$, $E = \{f(x, x) \approx a\}$. Assume now that $(\xleftrightarrow[E]{*})(L)$ is recognized by a tree automaton $\mathcal{A}$, wrt. the distinguished set of final states $\mathcal{S}$. We assume without loss of generality that $\mathcal{A}$ is deterministic. Since $\mathcal{A}$ has only finitely many states, there exist two distinct terms[1]

[1] $s^n(a) := \underbrace{s(\ldots s}_{n}(a))$

$s^{n_1}(a)$ and $s^{n_2}(a)$ which are recognized by $\mathcal{A}$ in the same state S. Since $f(s^{n_1}(a), s^{n_1}(a)) \in (\xleftrightarrow[E]{*})(L)$, and since $\mathcal{A}$ is deterministic, it contains a clause of the form $S(x_1), S(x_2) \parallel \;\rightarrow\; S_{\mathrm{f}}(f(x_1, x_2))$ where $S_{\mathrm{f}} \in \mathcal{S}$ is a final state, and thus $\mathcal{A} \models S_{\mathrm{f}}(f(s^{n_1}(a), s^{n_1}(a)))$. But this also implies $\mathcal{A} \models S_{\mathrm{f}}(f(s^{n_1}(a), s^{n_2}(a)))$ and this is a contradiction because this term $f(s^{n_1}(a), s^{n_2}(a))$ is not in $(\xleftrightarrow[E]{*})(L)$. □

3.2 Brothers automata and the non-linearities

Bogaert & Tison (1992) introduce tree automata with constraints which define a strict superclass of recognizable languages to deal with non-linear rewrite systems.

Definition 3.4
A tree automaton with equality constraints between brother subterms is a finite set of shallow declarations of the form $S_1(x_1), \ldots, S_n(x_n) \parallel \;\rightarrow\; S(f(x_1, \ldots, x_n))$ where the x_i are not necessarily distinct.

We call $Rec_=$ this class of recognizers as well as the class of recognized languages; the notion of recognized terms and languages is the same for tree automata with equality constraints between brother subterms as for (standard) tree automata.

The class $Rec_{\neq}$ (Bogaert & Tison 1992) is strictly larger than $Rec_=$ because (syntactic) disequations between variables $x_i \neq x_j$ are also allowed in the antecedent of clauses. The nice closure properties of tree automata still apply here, namely closure under Boolean operations, under determinism and complete specification. The emptiness problem is also decidable for $Rec_{\neq}$ though EXPTIME-complete (the problem of deciding emptiness of the intersection of some rational tree languages (see (Frühwirth, Shapiro, Vardi & Yardeni 1991, Seidl 1994) for a proof of EXPTIME-completeness) may indeed be reduced to emptiness for $Rec_{\neq}$). However, disequalities are not necessary for our purpose (see the conclusion for a discussion about this extension), but we can show that neither $Rec_=$ nor $Rec_{\neq}$ suffice to generalize Theorem 3.2 to the case of non-linear shallow systems.

Lemma 3.5
There exists some recognizable set L and (non-linear) shallow equational system E such that the set $(\xleftrightarrow[E]{*})(L)$ is not in $Rec_{\neq}$.

Proof. Let f, g be two binary function symbols, a be some constant and consider the system $E := \{f(x, x) \rightarrow g(x, x)\}$ and language $L := \{g(s_1, s_2) \mid s_1, s_2 \in \mathcal{T}(\mathcal{F})\}$. Assume that $L' := (\xleftrightarrow[E]{*})(L)$ is recognized by some $\mathcal{A} \in Rec_{\neq}$ with respect to the distinguished set of final states $\mathcal{S}$. We may assume without loss of generality that $\mathcal{A}$ is deterministic and completely specified. Let n be the number of states of $\mathcal{A}$.

We define a sequence of well-balanced ground terms of $\mathcal{T}(\mathcal{F})$ by $t_1 := f(a,a)$ and for all $i \geq 1$, $t_{i+1} := f(t_i, t_i)$. For all $i \geq 1$, the cardinality of the equivalence class of t_i modulo $\xleftrightarrow[E]{*}$ is 2^{2^i-1}. Let $i_0 = \lceil \log(\log(n+2)) \rceil$.

For each $i \geq i_0$, we have two distinct ground terms u_i and v_i, both equivalent to t_i modulo $\xleftrightarrow[E]{*}$ and both recognized by $\mathcal{A}$ in the same state called S_i. Moreover, by construction, $f(u_i, v_i) \in L'$. Thus, this term is recognized by $\mathcal{A}$ in some final state noted $T_i \in \mathcal{S}$. By determinism of $\mathcal{A}$, this means that there exists a clause $C_i = S_i(x_1), S_i(x_2) \parallel \rightarrow T_i(f(x_1, x_2)) \in \mathcal{A}^2$, such that $\mathcal{A} \models C_i\sigma$ with $\sigma = \{x_1 \mapsto u_i, x_2 \mapsto v_i\}$. Note that for all $i \geq i_0$, the variables x_1 and x_2 in clause C_i are distinct because $u_i \neq v_i$.

There exist two distinct integers $j > k \geq i_0$ such that $S_j = S_k$. Thus, $\mathcal{A} \models C_j\sigma$ where $\sigma = \{x_1 \mapsto u_j, x_2 \mapsto u_k\}$, because the variables $x_1 \neq x_2$ in C_j and thus $f(u_j, u_k)$ is recognized by $\mathcal{A}$ in the final state T_j. This is a contradiction because this term is not in L'. □

We can conclude from Lemma 3.5 that the syntactic equality constraints of the automata in $Rec_=$ are too rough for our purpose. The sorted shallow equational theories studied in the following section are a strict generalization of $Rec_=$. An important achievement of this approach is that semantic equality tests are possible.

3.3 Saturation

The following inference rules form schemata for sound and refutationally complete calculi for Horn clause sets consisting of declarations and sorted (dis)equations. They are mainly an adaption of basic superposition with selection (Bachmair et al. 1995, Nieuwenhuis & Rubio 1995) to the particular form of the Horn clauses considered here, where the sort constraints are always subject to the basic restriction and are solved by a particular selection strategy (Weidenbach 1996). This strategy is expressed by the rule Sort Constraint Resolution, see below. As usual, we assume a reduction ordering $\succ$ that is total on ground terms. We call the calculus consisting of the inference rules Basic Sort Constraint Resolution, Basic Superposition Right, Basic Superposition Left and Basic Equality Resolution plus the reduction rules subsumption and condensation (see Section 2) the *basic sorted superposition calculus*. The calculus consisting of the inference rules Basic Sort Constraint Resolution, Basic Paramodulation Right, Basic Paramodulation Left and Basic Equality Resolution plus the reduction rules subsumption and condensation the *basic sorted paramodulation calculus*. If term constraints are always eagerly propagated, i.e., we perform unification and apply the unifier to the literals, the resulting calculi are called the *sorted superposition calculus* and the *sorted paramodulation calculus*, respectively. Note that the

[2]Clauses defining rules of $Rec_{\neq}$ automata also contain syntactic disequations between variables, but these disequations do not matter here.

basic restriction does not interfere with subsumption or condensation, because we restricted the matchers in this context to only have variables in their codomain.

Definition 3.6 (Basic Sort Constraint Resolution)
The inference

$$
\text{Inf}\ \frac{\begin{array}{rl} T_1(t_1),\ldots,T_n(t_n),\Psi \parallel \Gamma \rightarrow \Delta & [\![P]\!] \\ \Theta_1 \parallel \quad \rightarrow T_1(s_1) & [\![Q_1]\!] \\ \vdots & \\ \Theta_n \parallel \quad \rightarrow T_n(s_n) & [\![Q_n]\!] \end{array}}{\bigcup_i \Theta_i, \Psi \parallel \Gamma \rightarrow \Delta \qquad [\![P \wedge (\bigwedge_i (Q_i \wedge t_i = s_i))]\!]}
$$

where no atom in Γ is selected, $\sigma = mgu(P)$,
  (i) $t_1\sigma = \ldots = t_n\sigma$ is a non-variable term
  or
  (ii) $t_i\sigma$ with $t_i\sigma \notin vars(\Gamma\sigma \cup \Delta\sigma)$,
no further atom $S(t_i\sigma)$ occurs in $\Psi\sigma$, and all Θ_i are solved, is called a *Basic Sort Constraint Resolution* inference.

Definition 3.7 (Basic Superposition/Paramodulation Right)
The inference

$$
\text{Inf}\ \frac{\begin{array}{rl} \Psi \parallel \rightarrow s \approx t & [\![P]\!] \\ \Theta \parallel \rightarrow A[s']_{1.p} & [\![Q]\!] \end{array}}{\Psi, \Theta \parallel \rightarrow A[1.p/t] \quad [\![P \wedge Q \wedge s' = s]\!]}
$$

where $t \not\succ s$, if A is an equation $l \approx r$ with $l|_p = s'$ then $r \not\succ l$, s' is not a variable, and the sort constraints Ψ and Θ are solved, is called a *Basic Superposition Right* inference. If we drop the requirement $r \not\succ l$ the inference is called a *Basic Paramodulation Right* inference.

Definition 3.8 (Basic Superposition/Paramodulation Left)
The inference

$$
\text{Inf}\ \frac{\begin{array}{rl} \Psi \parallel \qquad\qquad \rightarrow s \approx t & [\![P]\!] \\ \Theta \parallel l[s']_p \approx r \rightarrow & [\![Q]\!] \end{array}}{\Psi, \Theta \parallel l[p/t] \approx r \rightarrow \qquad [\![P \wedge Q \wedge s' = s]\!]}
$$

where $t \not\succ s$, $r \not\succ l$, s' is not a variable, and the sort constraints Ψ and Θ are solved or $l \approx r$ is selected, is called a *Superposition Left* inference. If we drop the requirement $r \not\succ l$ the inference is called a *Paramodulation Left* inference.

In case of a Basic Paramodulation Left inference the conclusion of the inference can be further refined to

$$\Psi, \Theta \parallel l[p/x] \approx r \rightarrow [\![P \wedge Q \wedge s' = s \wedge x = t]\!]$$

for some new variable x (Bachmair et al. 1995, Nieuwenhuis 1996).

Definition 3.9 (Basic Equality Resolution)
The inference

$$\text{Inf}\ \frac{\Theta \,\|\, s \approx t \rightarrow \ \llbracket P \rrbracket}{\Theta \,\| \qquad \rightarrow \ \llbracket P \wedge s = t \rrbracket}$$

where Θ is solved or $s \approx t$ is selected is called a *Equality Resolution* inference.

The next Lemma 3.10 shows that sorted shallow equational theories can be finitely saturated by sorted superposition. The process of exhaustively applying the inference rules of sorted superposition to the theory terminates in the sense that no new clauses are generated that are not redundant with respect to subsumption or condensation. Syntactically, this is always the case for some calculus, if the depth of terms in generated clauses as well as the cardinality of variable components of the clauses can be bound. Any set of clauses with respect to some finite number of predicate and function symbols where the depth of clauses as well as the cardinality of variable components of these clauses is bound by some constant and no clause in the set can be subsumed or condensed is finite. This characterization goes back to Joyner Jr. (1976).

Lemma 3.10
Sorted shallow equational theories can be finitely saturated by sorted superposition.

Proof. We shall show that the saturation process results in clauses of the form

$$T_1(t), \ldots, T_n(t), S_1(x_1), \ldots, S_m(x_m) \,\| \rightarrow A$$

where n, m are possibly zero, A is either a monadic atom $T(s)$ or an equation $l \approx r$ and t, s, l and r are always shallow terms. If the saturation process produces only clauses of this form, then it will terminate, because the depth of all these clauses as well as the length of variable chains between their literals are bound.[3] The length of variable chains is bound, because all predicates are monadic and there are at most three different non-variable shallow terms in any clause.

It remains to prove that all clauses generated by the saturation process have the above form. Obviously, shallow declaration clauses and sorted shallow equations are of the above form, where t as well as the x_i are variables occurring in A. For symmetry reasons it is sufficient to consider three cases of possible inferences: (i) The term t is a non-variable shallow term and we perform a sort constraint resolution inference. (ii) The term t is a variable that does not occur in A and we perform a sort constraint resolution inference. (iii) The sort constraint $T_1(t), \ldots, T_n(t), S_1(x_1), \ldots, S_m(x_m)$ is solved and we perform a superposition right inference. We separately consider

[3] See the discussion above this lemma.

these cases:
(i) The other clauses involved in the inference are all of the form $Q_1(y_1), \ldots, Q_{k_i}(y_{k_i}) \,\|\, \rightarrow T_i(s_i)$ where the y_j occur in s_i and s_i is a shallow term. The unifier σ only maps a variable to a non-variable shallow term if the variable is some s_i. Hence, the result of the inference is a clause of the desired form.
(ii) Again all other clauses involved in the inference are of the form $Q_1(y_1), \ldots, Q_{k_i}(y_{k_i}) \,\|\, \rightarrow T_i(s_i)$ where the y_j occur in s_i and s_i is a shallow term. The unifier σ possibly maps the variable t to a non-variable shallow term, but since t does not occur in A the result of the inference is again a clause of the desired form.
(iii) Since we do not superpose into variables and for any equation of the form $f(x_1, \ldots, x_n) \approx y$ either $y = x_i$ for some i and hence $f(x_1, \ldots, x_n) \succ x_i$ or y does not occur in $f(x_1, \ldots, x_n)$, a case analysis over the different combinations of the form of A and the involved sorted equation shows that the result is always of the desired form. Note that in the case of a Superposition Right inference, the involved clauses have a solved sort constraint. □

For example, we apply the saturation process to the sorted shallow equational theory presented in Section 1:

$$\begin{array}{lll}
(1) & S(x) & \| \rightarrow f(x,y) \approx h(y) \\
(2) & & \| \rightarrow f(x,x) \approx g(x) \\
(3) & & \| \rightarrow S(g(x))
\end{array}$$

where we assume $f(x,y) \succ g(x) \succ h(x)$. Then the saturation process generates the additional clauses (4) and (5) by Superposition Right inferences.

$$\begin{array}{lll}
(4) & S(x) & \| \rightarrow g(x) \approx h(x) \\
(5) & S(x) & \| \rightarrow S(h(x))
\end{array}$$

The clauses (1)–(5) are saturated by sorted superposition.

Corollary 3.11
Sorted shallow equational theories can be finitely saturated by sorted paramodulation.

Proof. The only difference between sorted superposition and sorted paramodulation is that the sorted paramodulation calculus paramodulates into both sides of equations, disregarding ordering restrictions. However, we only paramodulate with right hand sides of equations that are not smaller than the respective left hand side. Therefore, the proof of Lemma 3.10, in particular case (iii), carries over to sorted paramodulation. □

With respect to our example above, sorted paramodulation would produce the additional clause

$$(6) \quad S(x) \quad \| \to f(x,x) \approx h(x)$$

via a Paramodulation Right inference between the clauses (4) and (2). But clause (6) is subsumed by clause (1), hence the above clauses (1)–(5) are also saturated by sorted paramodulation.

Since our notion of subsumption and condensation is compatible with the basic restriction, the above results on finite saturation hold also for the basic variants of the sorted superposition/paramodulation calculus.

Corollary 3.12
Sorted shallow equational theories can be finitely saturated by basic sorted superposition and basic sorted paramodulation.

In particular, Corollary 3.12 allows for a very nice decidability proof of unification in sorted equational theories, since with respect to a clause set that is finitely saturated by basic sorted paramodulation, there are only finitely many applications of basic paramodulation left for some given equation. Therefore, we replaced the proof suggested in our conference paper (Jacquemard, Meyer & Weidenbach 1998) by the one below.

Lemma 3.13
Unifiability with respect to finitely saturated shallow sorted equational theories is decidable.

Proof. Two arbitrary terms t, s are unifiable iff we can derive the empty clause from the saturated theory and the goal clause $\| t \approx s \to$. Since the sorted shallow equational theory is saturated, no inferences inside the theory need to be considered. Furthermore, the goal is purely negative, so we can delete all clauses with an unsolved sort constraint from the saturated theory. Let us in particular assume that our theory is saturated by (basic) sorted paramodulation. Then by selecting the disequation $t \approx s$ only (basic) paramodulation left or equality resolution inferences can be performed until the disequation is resolved. Now for a theory saturated under (basic) sorted paramodulation, every position in the disequation has only to be considered once for a paramodulation left inference (see the remark below Definition 3.8). Hence, the application of Paramodulation Left and Equality Resolution terminates on the goal clause, resulting in a clause $S_1(t_1), \ldots, S_n(t_n) \| \to [\![P]\!]$. Now exhaustive application of Sort Constraint Resolution terminates on this clause, because the lexicographic combination of the number of different variables in the clause and the multiset of all term depths decreases with any Sort Constraint Resolution application. □

We evaluate two example queries with respect to the above saturated sorted shallow equational theory. First, we want to unify $f(x,y)$ and $h(y)$ starting with the goal clause

$$\| f(x,y) \approx h(y) \quad \to$$

We apply Superposition Left with (1) giving $S(x) \parallel z \approx h(y) \rightarrow [\![z = h(y)]\!]$. Next we apply Sort Constraint Resolution with (3) yielding $\parallel h(y) \approx h(y) \rightarrow [\![z = h(y)]\!]$ and finally an application of Equality Resolution yields the empty clause. Therefore, $f(x, y)$ and $h(y)$ are unifiable in the considered shallow equational theory.

Second, consider the unification problem of $f(a, x)$ and $h(x)$ where a is some constant. The problem has no solution justified by the saturated clause set consisting of the clauses (1)–(5) and the clauses below where we propagated all term constraints:

$$\begin{array}{rll} & \parallel f(a,x) \approx h(x) & \rightarrow \\ S(a) & \parallel & \rightarrow \\ & \parallel g(a) \approx h(a) & \rightarrow \end{array}$$

We already mentioned the close relationship between sorted unification, tree automata and the sort theories considered here in Section 3.1. This relationship can be exploited to derive results with respect to the number of unifiers of an equational problem.

Corollary 3.14 (Weidenbach (1998))
Let $\mathcal{L}$ be a finitely saturated shallow sorted equational theory and let $C = S_1(t_1), \ldots, S_n(t_n) \parallel \rightarrow$ be some clause. Then we can derive a clause $T_1(y_1), \ldots, T_k(y_k) \parallel \rightarrow$ from C and $\mathcal{L}$ by Sort Constraint Resolution iff the sorted unification problem $x_1 = t_1, \ldots, x_n = t_n$, $sort(x_i) = S_i$, has a well-sorted mgu with respect to the sort theory contained in $\mathcal{L}$ where all x_i are new.

Corollary 3.15 (Weidenbach (1998))
Unification in shallow sort theories is NP-complete, finitary and the number of well-sorted mgus is simply exponential in the size of the sort theory and unification problem.

Now from the above two corollaries and Lemma 3.13 we obtain that there are at most simply exponentially many well-sorted mgus for a unification problem with respect to a finitely saturated shallow sorted equational theory.

Corollary 3.16
The number of well-sorted mgus with respect to a finitely saturated shallow sorted equational theory is simply exponential. Deciding the unification problem is NP-hard.

The following example shows, that the number of syntactic mgus is already infinite for linear, shallow theories. Consider the saturated theory:

$$\begin{array}{rl} \parallel & \rightarrow S(a) \\ S(x) \parallel & \rightarrow S(g(x)) \\ \parallel & \rightarrow T(a) \\ T(x) \parallel & \rightarrow T(g(x)) \end{array}$$

where we can derive infinitely many syntactic mgus of the form $\{x \mapsto g^i(a)\}$ from the clause $S(x), T(x) \parallel \rightarrow$.

4 Semi-Linear Sorted Equational Theories

In this section we prove that unification in semi-linear equational theories is decidable, too. We do so by transforming a semi-linear equational theory into a sorted shallow equational theory, preserving satisfiability. Then we apply Lemma 3.10 and Lemma 3.13 to obtain the decidability result. The following rule transforms sorted semi-linear equational theories into sorted shallow equational theories.

Definition 4.1
The transformation

$$\text{Red}\ \frac{\Psi, S_1(x_1), \ldots, S_m(x_m) \parallel \rightarrow A[t]_{p_1}}{\begin{array}{r} S_1(x_1), \ldots, S_m(x_m) \parallel \rightarrow T(t) \\ T(x), \Psi \parallel \rightarrow A[p_1, \ldots, p_n/x] \end{array}}$$

provided t is a non-variable subterm, $x_i \in vars(t)$ for all i, $vars(\Psi) \cap vars(t) = \emptyset$, $|p_i| = 2$ for all i, the positions $p_1, \ldots, p_n$ refer to all positions q of t in A with $|q| = 2$, T is a new monadic predicate and x is new to the replaced clause, is called *flattening.*

Lemma 4.2
Exhaustive application of flattening to a (sorted) semi-linear equational theory terminates, results in a sorted shallow equational theory and preserves satisfiability.

Proof. Termination follows from the fact that the transformation replaces a clause by two clauses with strictly fewer function symbols. No transformation is applicable to a clause that is a shallow declaration or a sorted shallow equation, since all terms at depth two of such atoms are always variables (if they exist). On the other hand, if the direct subterm of an atom is not shallow, it has a subterm at depth two which is not a variable and therefore the transformation applies as long as the transformation preserves semi-linearity (see below). Hence, the transformation terminates in a sorted shallow equational theory.

By an induction argument it is sufficient to show that a single step of the transformation preserves satisfiability and results in a sorted semi-linear equational theory. The crucial property is that $vars(t) \cap vars(A[p_1, \ldots, p_n/x]) = \emptyset$. We show this by contradiction. Assume that after an application of the transformation there is a variable y occurring in t and $A[p_1, \ldots, p_n/x]$. By construction this can only be the case if y has an occurrence in A that is not inside an occurrence of t in A. So y occurs in some term $s \neq t$ with $A|_q = s$, $|q| = 2$, contradicting that the original clause was semi-linear. For the same reason, the result of an application of the transformation is again a sorted semi-linear equational theory and $x_j \notin vars(A[p_1, \ldots, p_n/x])$ for all j. $\square$

Theorem 4.3
Unifiability in semi-linear equational theories is decidable.

Proof. By Lemma 4.2 we can effectively translate semi-linear equational theories into sorted shallow equational theories preserving satisfiability. By Lemma 3.10 these theories can be effectively saturated by sorted superposition and by Lemma 3.13 unifiability is decidable with respect to saturated sorted shallow equational theories. $\square$

Application of the transformation to the example presented in the introduction yields the sorted shallow equational theory considered in the previous section.

4.1 Applications

Any equational theory E can be transformed into a semi-linear equational theory E' by replacing non-linear variable occurrences with fresh variables. Then E' is an upper approximation for E in the sense that $\overset{*}{\underset{E}{\leftrightarrow}} \subseteq \overset{*}{\underset{E'}{\leftrightarrow}}$, i.e., non-unifiability in E' implies non-unifiability in E. Furthermore, by Theorem 4.3, non-unifiability in E' is decidable. Ganzinger, Meyer & Weidenbach (1997) showed that in this case non-unifiability in E' can be used to effectively direct the search of a theorem prover in finding proofs with respect to E. One of our future goals is to improve the performance of SPASS (Weidenbach 1997, Weidenbach, Meyer, Cohrs, Engel & Keen 1998) using this technology. Note that flattening applied to an arbitrary equational theory where we keep some $S_i(x_i)$ in the transformed clause if $x_i \in vars(A[p_1, \ldots, p_n/x])$ is already a transformation that generates an appropriate approximation.

5 Extensions

A possible extension is to apply our method to compute the (eventual) solution of a unification problem in a semi-linear theory. Weidenbach (1996) showed that sort constraint resolution simulates sorted unification. Unification in shallow sort theories is known to be NP-complete and of unification type finitary. This implies that unification in sorted shallow equational theories is NP-hard and also of unification type finitary, if we consider well-sorted unifiers. The results of Theorem 4.3 in Section 4 obviously extend to sorted semi-linear equational theories.

Generalization of standard theories. The standard equations in (Nieuwenhuis 1996) include one form which is not embedded by the semi-linear case: the form $f(\ldots, g(x), \ldots) \approx x$ where g is a unary function symbol, assuming additional restrictions on the positions of linear terms and non-linear shallow variables in other equations. Obviously, the subterm $g(x)$ cannot be transformed into a sort declaration. However, we can show that E-unification in those theories can still be decided by basic sorted paramodulation. The following discussion mainly follows the previous argumentation on sorted shallow and semi-linear systems. Additionally, we slightly generalize the form of invariants of the saturation process. Moreover, our definition of shallow terms has to be adapted such that not only variables but also constants as arguments of shallow terms are allowed. In the sequel we will use this generalized definition of shallow terms.

In order to show that the generalized fragment is still closed under basic sorted paramodulation we have to treat certain equations which contain variable disjoint sides carefully, e.g. the equation $f(x,y) \approx g(z)$. Equations of this form cannot be oriented by any admissible ordering and thus have to be applied in both directions. Saturation may not terminate when such equations are superposed on subterms of the form $g(x)$ in an equation $f(\ldots, g(x), \ldots) \approx x$. Arbitrary deep term structures can be generated by subsequent paramodulation steps. However, it is possible to transform such equations dynamically during the saturation process. For example, the equation $f(x,y) \approx g(z)$ can be transformed into equations $f(x,y) \approx a$ and $g(z) \approx a$ where a is a new constant symbol. As a consequence we have to show that only a finite number of new symbols can be generated.

Consider the simplest equation $x \approx y$. If this equation is derived we can just stop saturation since the Herbrand universe collapses in this case. For more complicated sorted equations with variable disjoint sides, e.g. $S(x) \,\|\, \rightarrow x \approx t$ and $x \notin \mathit{vars}(t)$, we can only deduce similar information about the particular sort S. As we have mentioned above a critical situation may come up when such equations are superposed onto a subterm $g(x)$. In general, this may happen using sorted equations of the form $S(x), \Psi \,\|\, \rightarrow x \approx t$ or $S(x), \Psi \,\|\, \rightarrow g(x) \approx t$ where $x \notin \mathit{vars}(t)$. Importantly, equations of the form $\Psi \,\|\, \rightarrow f_1(x_1, \ldots, x_n) \approx f_2(y_1, \ldots, y_m)$ where $\{x_1, \ldots, x_n\} \cap \{y_1, \ldots, y_m\} = \emptyset$ and $n, m > 1$ do not have to be considered. Paramodulation by those equations onto a subterm $g(x)$ is not possible.

Consider the transformation of a sorted equation $S(x) \,\|\, \rightarrow x = f(y,y)$ into two equations $S(x) \,\|\, \rightarrow x = a$ and $S(x) \,\|\, \rightarrow f(y,y) = a$. Such a transformation step is called *splitting*. The splitting of similar unsorted equations has already been used in (Nieuwenhuis 1996). If the constant symbol a is a new symbol splitting clearly preserves (un-)satisfiability. However, (un-)satisfiability is also preserved if the constant a is being reused if a has been introduced before for the same solved sort constraint $S(x)$.

Splitting terminates for clause sets which contain only finitely many sorts. There are two important observations. For finitely many sorts there are an exponential number of intersections of sorts. Thus variables can have only finitely many different sorts. Shallow equations in which both sides have root symbols with arity greater than one are not split. For if one side contains a newly introduced constant symbol as an argument splitting would generate new constant symbols which depend on previously introduced symbols. For example, splitting of a sorted equation $S(x) \parallel \rightarrow f(x, a) = h(y, z)$ generates a sorted equation $S(x) \parallel \rightarrow f(x, a) = b$ where a is a constant symbols previously introduced by splitting and b is a new constant symbol. In this way splitting does not terminate in general.

Despite of the syntactic flavor of splitting we can also motivate it semantically. For the equation $S(x) \parallel \rightarrow x = f(z, z)$ to become true the sort S contains at most one element. Splitting introduces an explicit name for this element, if it exists.

Definition 5.1
Let N be a set of clauses. A splitting step is defined as the transformation of a clause in N by the rule *Split1* or *Split2*:

$$\text{Split1}\ \frac{\Psi \parallel \rightarrow x \approx t}{\begin{array}{c}\Psi \parallel \rightarrow x \approx a \\ \Psi \parallel \rightarrow t \approx a\end{array}}$$

where the sort constraint Ψ is solved, t is a shallow term, the top symbol of t is not a unary function symbol, $x \notin \mathit{vars}(t)$, and if there is a clause $\Psi \parallel \rightarrow x \approx b$ in N then let $a := b$ or, otherwise, let a be a fresh constant symbol.

$$\text{Split2}\ \frac{\Psi \parallel \rightarrow g(x) \approx t}{\begin{array}{c}\Psi \parallel \rightarrow g(x) \approx a \\ \Psi \parallel \rightarrow t \approx a\end{array}}$$

where the sort constraint Ψ is solved, t is a shallow term, g is a unary function symbol, $x \notin \mathit{vars}(t)$, and if there is a clause $\Psi \parallel \rightarrow g(x) \approx b$ in N then let $a := b$ or, otherwise, let a be a fresh constant symbol.

Proposition 5.2
Let N be a set of clauses of the following forms:

1. $\Psi \parallel \rightarrow x \approx a$ or
2. $\Psi \parallel \rightarrow g(x) \approx a$

where the sort constraint Ψ is solved, a is a constant, and g is a unary function symbol. Then up to different a, variable renaming, and condensing the set N is finite.

Proof. Let n be the number of different monadic predicate symbols and let m be the number of unary function symbols, occurring in clauses of N. Then there are 2^n different sort constraints on one variable and $O(2^n)$ different sort constraints on several variables up to condensing. Consequently, the set N contains $O(2^n)$ different clauses up to different a, variable renaming, and condensing. □

In the following lemma we show that splitting can only be applied finitely many times. Sorted equations are split only if at least one side is a sorted variable or a term $g(x)$ where g is unary and x is a sorted variable. More complex sorted equations $f(s_1, \ldots, s_n) \approx h(t_1, \ldots, t_m)$ are not split if f and h are n-ary functions with $n > 1$ and the s_i and t_j are variables or constants. Recursive occurrences of formerly introduced constants would lead to an infinite splitting process. A constant introduced by splitting is a name for the single element in a particular intersection of sorts on one variable. In this sense constants are reused by splitting whenever the same intersection of sorts arises several times. Finally, splitting is applied only to sorted equations in which both sides are shallow. Obviously, there are only finitely many different sorted shallow terms over a finite signature.

Lemma 5.3
Let N be a set of clauses over a finite signature $\mathcal{F}$. Then splitting can be applied only finitely many times in any derivation of the basic ordered paramodulation calculus with splitting.

Proof. Only the splitting rules Split1 and Split2 may introduce new constant symbols. We assume that all sort constraints are minimal in the sense that no further condensing is possible. Let C be a clause $\Psi \parallel \rightarrow x \approx t$ from N where the conditions of Split1 hold for C. Suppose that there is no clause $\Psi \parallel \rightarrow x \approx b$ in N. In this case Split1 introduces a new constant symbol a and transforms C to $C' = \Psi \parallel \rightarrow x \approx a$. Note that in general the sort constraint Ψ is not in solved form in C' anymore since the equation does not contain the variables of t. Suppose the sort constraint of C' has been simplified to Ψ' where Ψ' does not need to be in solved form. The constant a may be viewed as a label for Ψ', not for Ψ. Thus in a subsequent application of Split1 to a clause $\Psi \parallel \rightarrow x \approx t'$ a new constant has to be introduced. However, an application of Split1 to a clause $\Psi' \parallel \rightarrow x \approx t'$ can reuse the constant a. By Proposition 5.2 there are only finitely many clauses of the form $\Psi \parallel \rightarrow x \approx a$. Thus Split1 introduces only finitely many new constant symbols. Note that there are only finitely many different sorted shallow terms over a finite signature up to variable renaming and condensing. Thus Split1 can even be applied only finitely many times. The proof for Split2 is similar. □

Lemma 5.4
Let N and N' be two sets of clauses. The set N' is the result of a transformation of N by the rules Split1 and Split2. Then N is satisfiable if and only if N' is satisfiable.

Proof. Let N be a set of clauses and let C be the clause $\Psi \,\|\, \rightarrow x \approx t$ in N where the conditions of Split1 hold for C. Let C' and D be the conclusions $\Psi \,\|\, \rightarrow x \approx a$ and $\Psi \,\|\, \rightarrow t \approx a$, respectively, of an application of Split1 to C. Let N' be the set $\{N \setminus C\} \cup \{C', D\}$.

Suppose I is a model of N. If there is no clause $\Psi \,\|\, \rightarrow x \approx b$ in N then a is a fresh constant symbol and we may construct I' by adding, for all ground substitutions σ such that $\Psi\sigma \subseteq I$, the ground instances $x\sigma \approx a$ and $t\sigma \approx a$ to I. Note that for any ground substitution σ with $\Psi\sigma \subseteq I$ we have that $x\sigma = t\sigma$ is true in I since C is true in I. Thus I' is a model of N'.

Let C'' be a clause $\Psi \,\|\, \rightarrow x \approx b$ in N. In this case a has been chosen to be equal to b. Since C'' is true in I the clause C' is also true in I. To show that D is true in I we distinguish two cases. Suppose that σ is a ground substitution where $\Psi\sigma \subseteq I$. Then $x\sigma \approx t\sigma \in I$ and $x\sigma \approx b \in I$ since C and C'' are true in I, respectively. By the transitivity of $\approx$ we have that $t\sigma \approx b$ and thus $D\sigma$ are true in I. On the other hand, suppose that for a ground substitution σ there is a sort constraint $A \in \Psi$ where $A\sigma \notin I$. Then $D\sigma$ is also true in I. Consequently, I is a model of N'.

The other direction follows immediately since C is a consequence of a (sound) paramodulation step on a in C' to t using D. The proof for Split2 is similar. □

Note that we can also show that splitting is monotone in the sense that the conclusions are always smaller then the premise assuming that constant symbols are smaller in the precedence than any non-constant function symbol. Suppose the potential splitting candidate $S(x), \Psi \,\|\, \rightarrow x \approx t$ is transformed using the constant symbol a. Then all instances of the premise where x is substituted by a constant b and b is smaller than a have to be added. Similarly, for a splitting candidate $S(x), \Psi \,\|\, \rightarrow g(x) \approx t$ where t is a variable all instances with t substituted by a constant b have to be enumerated. We may assume that there are only finitely many constants b which are smaller than a. Under these assumptions splitting remains effective and is an admissible simplification rule, c.f. (Bachmair et al. 1995).

Lemma 5.5
The combination of the basic ordered paramodulation calculus with splitting is sound and refutationally complete assuming a finite signature $\mathcal{F}$.

Proof. By Lemma 5.4 splitting preserves (un-)satisfiability. We may additionally assume that splitting is an admissible simplification rule in the sense of Bachmair et al. (1995), see the discussion above. Due to Lemma 5.3 splitting can be applied only finitely many times and thus fairness is not affected. Consequently, the lemma follows from the soundness and refutational completeness of basic ordered paramodulation. □

We call an equation $f(t_1, \ldots, t_n) \approx x$ *semi-shallow* if for all i with $1 \leq i \leq n$ the term t_i is either a variable or the term $g(x)$ where g is a unary function symbol. Thus several occurrences of $g(x)$ are possible. If there is any $g(x)$ among the t_i then all other t_j which are variables are distinct from x. A sorted equational theory is called *semi-shallow* if all declarations are shallow and all equations are either shallow or semi-shallow where each semi-shallow equation $f(t_1, \ldots, t_n) \approx x$ contains at most one t_i of the form $g(x)$. An equation is called *collapsing* if at least one side of the equation is a variable. Otherwise, the equation is called *non-collapsing.*

We assume that equality constraints are propagated eagerly into the clause part unless a semi-shallow equation is generated. For example, a clause $S(x) \,\|\, \rightarrow x \approx y[\![x = g(y)]\!]$ is transformed into $S(g(y)) \,\|\, \rightarrow g(y) \approx y[\![\top]\!]$ whereas a clause $S(x) \,\|\, \rightarrow f(x) \approx y[\![x = g(y)]\!]$ is transformed into $S(g(y)) \,\|\, \rightarrow f(x) \approx y[\![x = g(y)]\!]$. The constraint propagation corresponds to a weakening of the equational constraint which is always possible, c.f. (Nieuwenhuis & Rubio 1995).

Note that in the following lemma we do not explicitly include constants as arguments of shallow terms in the definition of the invariant, although constants may also occur in this way during saturation. However, as long as constants are the smallest symbols in the precedence, constants can only be replaced by constants using paramodulation. Therefore, we skip this case to simplify technical matters.

Lemma 5.6
Sorted semi-shallow equational theories over a finite signature $\mathcal{F}$ can be finitely saturated by basic sorted paramodulation and splitting.

Proof. The closure process is parameterized by a simplification ordering $\succ$ on terms assuming that the precedence $\succ_p$ is compatible with arities, i.e. whenever the arity of a function symbol f is greater than the arity of a function symbol g then $f \succ_p g$. Note that therefore constants are the smallest symbols in the precedence. Additionally, we assume that splitting is done with the highest priority, i.e. a sorted equation is always split if possible before any other rule is applied to this equation.

We show that saturation under basic sorted paramodulation with splitting is closed for sets of clauses of the following forms:

1. $S_1(s), \ldots, S_n(s), T_1(t), \ldots, T_m(t), \Psi \,\|\, \rightarrow A[\![\top]\!]$ where n, m are possibly zero, Ψ is solved, A is either a monadic atom $P(t')$ or an equation $l \approx r$, and s, t, t', l, and r are always shallow terms.

2. $\Psi \parallel \rightarrow f(x_1, \ldots, g(x), \ldots, x_n) \approx x[\![\top]\!]$ where n is possibly zero, Ψ is solved, g is a unary function symbol, and $g(x)$ occurs only once.

3. $T_1(t), \ldots, T_n(t), \Psi \parallel \rightarrow f(x_1, \ldots, x_m) \approx x[\![x_{i_1} = g(x), \ldots, x_{i_k} = g(x)]\!]$ where n and m are possibly zero and $k > 0$, Ψ is solved, g is a unary function symbol, $x \notin \{x_1, \ldots, x_m\}$, and t is a shallow term.

Each clause of the original theory belongs to Category 1 or Category 2 and has a solved sort constraint. The predicates in these clauses are all predicates of the original theory.

Assuming a finite signature there is only a bounded (with respect to the theory) number of clauses which belong to each of the three categories because the depth of all these clauses as well as the length of variable chains between its literals is bounded. Hence, there are only finitely many different clauses of this form with respect to subsumption and condensing, see Section 2. If the saturation process produces only clauses of this form, it will terminate.

If a rule in Category 1, 2, or 3 has unsolved sort constraints then we can perform a basic sort constraint resolution inference. The result is a clause of the same category. Otherwise, there are several ways to apply basic paramodulation right on clauses with solved sort constraints:

1. The basic paramodulation right of a clause C_1 in Category 1 on a clause C_2 in Category 1 results in a clause C_3 of the same category shown in the proof of Lemma 3.10. Note that this result still holds for the extended definition of shallow terms since constants are the smallest symbols in the precedence.

2. The basic paramodulation right of a clause C_1 in Category 1 on a clause C_2 in Category 2 results in a clause C_3 of:

   (a) Category 3 if C_1 contains a non-collapsing equation and is applied at topmost position.

   (b) Category 2 if C_1 is a non-collapsing sorted equation $\Theta \parallel \rightarrow g(x) \approx h(x)[\![\top]\!]$ and is applied on non-topmost position. Note that an equation of the form $f(s_1, \ldots, x, \ldots, s_k) \approx g(x)$ where $k > 1$ cannot be applied because it is always oriented from left to right due to the assumption that the precedence $\succ_p$ is compatible with the arity of function symbols.

   (c) Category 1 if C_1 is a sorted equation $\Theta \parallel \rightarrow g(x) \approx t[\![\top]\!]$ or $\Theta \parallel \rightarrow x \approx t[\![\top]\!]$ where $x \notin \mathit{vars}(t)$ and is applied at non-topmost position. Splitting ensures that t is always a constant. Note that the equation in C_3 may contain constants as arguments. However, we do not consider this case explicitly here to simplify technical matters. Informally, constants can only be paramodulated to other constants since constant symbols are assumed to be the smallest symbols in the precedence. Thus constants are similar to variables in this case.

(d) Category 1 if C_1 contains a collapsing equation, regardless of topmost or non-topmost application.

3. The basic paramodulation right of a clause C_1 in Category 2 on a clause C_2 in Category 1 results in a clause C_3 of:

   (a) Category 3 if C_2 contains a non-collapsing equation. If the equality constraint can be propagated without getting a semi-shallow equation then C_3 belongs to Category 1.

   (b) Category 1 if C_2 contains a collapsing equation or a sort declaration.

4. The basic paramodulation right of a clause C_1 in Category 2 or 3 on a clause C_2 in Category 2 or 3 results in a clause C_3 of Category 1. Note that if both C_1 and C_2 belong to Category 2 then two different non-variable terms may occur in the sort constraint of the resulting clause C_3 of Category 1. For example, if $C_1 = S_1(y) \parallel \rightarrow f(g(x), y) \approx x[\![\top]\!]$ and $C_2 = S_2(z) \parallel \rightarrow f(z, h(u)) \approx u[\![\top]\!]$ then $C_3 = S_1(h(u)), S_2(g(x)) \parallel \rightarrow u \approx x[\![\top]\!]$. Note that C_3 will be split if the sort constraint of C_3 is solvable. Another example explains the restriction of semi-shallow theories where only one occurrence of a term $g(x)$ is allowed in each semi-shallow equation. Suppose $C_1 = \parallel \rightarrow f(g(x), g(x)) \approx x[\![\top]\!]$ and $C_2 = \parallel \rightarrow g(h(y)) \approx y[\![\top]\!]$ where C_1 is obviously semi-shallow but is not allowed in the original clause set. Then we may derive $C_3 = \parallel \rightarrow f(y, g(h(y))) \approx h(y)[\![\top]\!]$ which is also not semi-shallow.

5. The basic paramodulation right of a clause C_1 in Category 3 on a clause C_2 in Category 1 or vice versa results in a clause C_3 of:

   (a) Category 3 if C_2 contains a non-collapsing equation. If the equality constraint can be propagated without getting a semi-shallow equation then C_3 belongs to Category 1.

   (b) Category 1 if C_2 contains a collapsing equation or a sort declaration.

It is important to see that clauses of Category 3 which contain several occurrences of a subterm $g(x)$ appear only during the saturation process. The occurrences of the $g(x)$ in those clauses are blocked due to basic restrictions. In other words, a paramodulation step on a subterm $g(x)$ is only possible in clauses of Category 2. Those clauses either belong to the original theory or are conclusions of case 2b. □

Lemma 5.7
Unifiability with respect to finitely saturated sorted semi-shallow equational theories is decidable.

Proof. The result follows from the proof of Lemma 3.13 which can easily be extended to sorted semi-shallow equational theories. □

We call an equation $f(t_1, \ldots, t_n) \approx x$ *semi-standard* if $f(t_1, \ldots, t_n)$ is semi-linear and moreover, there is one unary symbol g such that for all t_i with $x \in \mathit{vars}(t_i)$ we have that $t_i = g(x)$. An equational theory E is called *semi-standard* if E only contains semi-linear equations or semi-standard equations of the form $f(t_1, \ldots, t_n) \approx x$ where only one t_i can be of the form $g(x)$.

Note that similar to the semi-linear case a semi-standard equational theory can be transformed into a sorted semi-shallow equational theory while satisfiability is preserved. The transformation procedure of a semi-linear theory into a sorted shallow system due to Lemma 4.2 can easily be extended to work also for semi-standard theories.

It is sufficient for this purpose to leave the occurrences of subterms of the form $g(x)$ untouched during the transformation. The transformation is performed by using the following rules for the transformation of (sorted) equations and declarations. Since there are no equality constraints in clauses of the original theory we use the standard clause notation.

Definition 5.8
The following transformation is called *semi-flattening*:

$$\text{Red}\ \frac{\Psi, S_1(x_1), \ldots, S_m(x_m) \parallel \rightarrow A[t]_{p_1}}{\begin{array}{r} S_1(x_1), \ldots, S_m(x_m) \parallel \rightarrow T(t) \\ T(x), \Psi \parallel \rightarrow A[p_1, \ldots, p_n/x] \end{array}}$$

provided t is a non-variable subterm, $x_i \in \mathit{vars}(t)$ for all i, $\mathit{vars}(\Psi) \cap \mathit{vars}(t) = \emptyset$, $|p_i| = 2$ for all i, the positions $p_1, \ldots, p_n$ refer to all positions q of t in A with $|q| = 2$, T is a new monadic predicate, x is new to the replaced clause, and if A is a semi-standard equation $s \approx y$ then y does not occur in t.

Lemma 5.9
Exhaustive application of the transformation presented in Definition 5.8 to a semi-standard equational theory terminates, it results in a sorted semi-shallow equational theory and it preserves satisfiability.

Proof. The transformation procedure is almost the same as the one of Definition 4.1 where an occurrence of $g(x)$ in a semi-standard equation $f(\ldots, g(x), \ldots) \approx x$ is left untouched. Consequently, the transformation terminates and preserves satisfiability, c.f. Lemma 4.2. □

Theorem 5.10
Unifiability in semi-standard equational theories is decidable.

Proof. By Lemma 5.9 we can translate E into a sorted semi-shallow equational theory while satisfiability is preserved. Now by Lemma 5.6 such a theory can be effectively saturated by basic sorted paramodulation and splitting. Due to Lemma 5.7 unifiability is decidable with respect to finitely saturated sorted semi-shallow equational theories. □

6 Limitations and Related Work

6.1 Limitations

We present a generalization of semi-linear equational systems which cannot be treated with the methods of Sections 3.3 and 4.

The combination of associativity for one function symbol and a linear (!) shallow sort theory already yields an undecidable unification problem. This can be seen by a reduction of the emptiness of the intersection of context free languages to this problem.

Also, if the syntactic conditions in the definition of semi-linear systems are weakened, then unifiability becomes undecidable. A notion of *pseudo-linear* sort theories is introduced in (Weidenbach 1996) as a generalization of semi-linear sort theories in which unification is still decidable. However, our saturation method does not permit to solve unification in the corresponding pseudo-linear equational theories. These theories generalize sorted semi-linear equational theories in a way that multiple occurrences of a variable in an equation are allowed, provided they are all at the same depth. For instance, every semi-linear theory is pseudo-linear, $f(h(x), g(x)) \approx g(g(x))$ and $m(h(x), g(x), y) \approx y$ are pseudo-linear equations which are not semi-linear and $f(h(x), g(x)) \approx g(x)$ is not pseudo-linear.

Proposition 6.1
The word problem for pseudo-linear equational theories is undecidable. Even for theories in which all symbols are unary (string equational systems), and equations are ground or have one of the forms $a(b(x)) \approx c(d(x))$ or $a(x) = t$ and t is ground.

This kind of theory is close to the simplest case of pseudo-linear non-shallow equational system. The Proposition 6.1 characterizes a gap between semi-linear and pseudo-linear systems with respect to the transformation presented in Section 4.

Proof. (proposition 6.1) We reduce the Turing machine blank accepting problem. Let $\mathcal{M}$ be a Turing machine with input alphabet $\{0, 1, B\}$ (B is the blanc symbol), with states set Q and with transition function

$\delta : \{0, 1, B\} \times Q \to \{0, 1, B\} \times Q \times \{\text{Left}, \text{Right}\}$. Note that we do not use marker symbols. The B will actually replace the markers. We make the following non-restrictive assumptions concerning $\mathcal{M}$:

1. $\mathcal{M}$ is deterministic (i.e. δ is indeed functional);
2. The tape of $\mathcal{M}$ has always the form of a word of $B^\infty\{0,1\}^*B^\infty$ during computations (B^∞ represents an infinite sequence of blanc symbols);
3. $\mathcal{M}$ starts its computations in the state $q_{\text{in}} \in Q$;
4. $\mathcal{M}$ always stops its successful computations by cleaning (replacing by B) all the non-B symbols of the tape and then goes to the unique final state q_{f};
5. Moreover, we assume that no move is possible if $\mathcal{M}$ is in state q_{f};

To ensure that $\mathcal{M}$ does not violate the condition 2, we can add to $\mathcal{M}$ the ability to make few moves close to the position of the head (while letting the symbols on tape untouched) in order to check a replacement of an B by a 0 or an 1 or vice-versa is conform to condition 2. Lets consider the following alphabets:

$$A_{\mathcal{M}} = \{\epsilon, 0, 1, B\} \uplus \{0_q \mid q \in Q\} \uplus \{1_q \mid q \in Q\} \uplus \{B_q \mid q \in Q\}$$

$$A = A_{\mathcal{M}} \uplus \{T, T'\}$$

All symbols of A have arity 1 except ϵ which is a constant. For sake of simplicity, we note $a_1a_2\ldots a_n(\epsilon)$ (or $a_1a_2\ldots a_n(x)$, $x \in \mathcal{X}$) for $a_1(a_2(\ldots a_n(\epsilon)))$ (resp. $a_1(a_2(\ldots a_n(x))))$. As usual, we make no distinction between a word $a_0a_1\ldots a_n \in (A \setminus \{\epsilon\})^*$ and the ground term $a_1(a_2(\ldots a_n(\epsilon))) \in \mathcal{T}(A)$. In particular, we may use regular expressions (star notation) to design subsets of $\mathcal{T}(A)$.

The configurations of $\mathcal{M}$ will be represented by words of $B^*A_{\mathcal{M}}^*B^*$. The aim of symbols of the form 0_q, 1_q or B_q is to represent both the state and the position of the head of $\mathcal{M}$ in a configuration. More precisely, assume that $\mathcal{M}$ is in state q and has $B^\infty a_1\ldots a_nB^\infty$ on the tape and its head is at position $i \leq n$. Then we represent this configuration by a word of $B^*a_1a_2\ldots a_{i-1}a_qa_{i+1}\ldots a_nB^*$, where $a = a_i$.

In a first step in the reduction of the blank accepting problem for $\mathcal{M}$, we construct a *term rewriting system* (TRS) $\mathcal{R}$ instead of an equational system. In the next step the rewrite rules of $\mathcal{R}$ will be unoriented.

A (pseudo-linear) sub-system of $\mathcal{R}$ called $\mathcal{R}_{\mathcal{M}}$ is associated to the transition rules of $\mathcal{M}$. It is defined on $A_{\mathcal{M}}$ in the following manner: for each $(a, q) \in \{0, 1, B\} \times Q$,

- If $\delta(a, q) = (b, q', \text{Left})$, then
  $\mathcal{R}_{\mathcal{M}} \ni a'a_q(x) \to a'_{q'}b(x)$ for every $a' \in \{0, 1, B\}$

- If $\delta(a,q) = (b, q', \text{Right})$, then
  $\mathcal{R}_\mathcal{M} \ni a_q a'(x) \to b a'_{q'}(x)$ for every $a' \in \{0, 1, B\}$.

Lemma 6.2
$\mathcal{M}$ accepts a blank tape B^∞ iff $\{v \in B^* B_{q_\mathrm{f}} B^* \mid \exists u \in B^* B_{q_\mathrm{in}} B^*,\ u \xrightarrow[\mathcal{R}_\mathcal{M}]{*} v\} \neq \emptyset$

The words of $B^* B_{q_\mathrm{f}} B^*$ are generated from $T(\epsilon)$ by the ground TRS $\mathcal{R}_1$.

$$\mathcal{R}_1 = \left\{ \begin{array}{lclclcl} T(\epsilon) & \to & BT(\epsilon) & & T(\epsilon) & \to & B_{q_\mathrm{in}} T'(\epsilon) \\ T'(\epsilon) & \to & BT'(\epsilon) & & T'(\epsilon) & \to & C(\epsilon) \end{array} \right\}$$

The third shallow TRS $\mathcal{R}_2$ reduces a ground term of $B^* B_{q_\mathrm{f}} B^* C$ to $B_{q_\mathrm{f}}(\epsilon)$

$$\mathcal{R}_2 = \{B_{q_\mathrm{f}}(x) \to B_{q_\mathrm{f}}(\epsilon) \quad BB_{q_\mathrm{f}}(x) \to B_{q_\mathrm{f}}(\epsilon)\}$$

Let $\mathcal{R} = \mathcal{R}_1 \cup \mathcal{R}_\mathcal{M} \cup \mathcal{R}_2$.

Lemma 6.3
The Turing machine $\mathcal{M}$ accepts a blank tape iff $T(\epsilon) \xrightarrow[\mathcal{R}]{*} B_{q_\mathrm{f}}(\epsilon)$.

Proof. By construction of $\mathcal{R}_1$ and $\mathcal{R}_2$ and by Lemma 6.2, the existence of a rewriting sequence of the form $T(\epsilon) \xrightarrow[\mathcal{R}_1]{*} u \xrightarrow[\mathcal{R}_\mathcal{M}]{*} v \xrightarrow[\mathcal{R}_2]{*} B_{q_\mathrm{f}}(\epsilon)$ (with $u, v \in \mathcal{T}(A)$) is equivalent of the blank accepting for $\mathcal{M}$.

Also, any rewriting $T(\epsilon) \xrightarrow[\mathcal{R}]{*} B_{q_\mathrm{f}}(\epsilon)$ can be transformed into a sequence of the above form by permutations. Indeed, $\mathcal{R}$ has no critical pairs by hypothesis on $\mathcal{M}$ (Hypothesis 5) and by construction of $\mathcal{R}_\mathcal{M}$. □

We let $E = \{l \approx r \mid l \to r \in \mathcal{R}\}$. Since $\mathcal{R}$ has no critical pairs and $B_{q_\mathrm{f}}(\epsilon)$ is an $\mathcal{R}$-normal-form, we have:

Lemma 6.4
The Turing machine $\mathcal{M}$ accepts a blank tape iff $T(\epsilon) \xleftrightarrow[E]{*} B_{q_\mathrm{f}}(\epsilon)$.

□

Note that this reduction also proves that $(\xleftrightarrow[E]{*})(L)$ is not necessarily recognizable when L is recognizable and E is a pseudo-linear equational system.

A reduction of the blank accepting problem to a unifiability problem for a pseudo-linear string equational system in which all equations have the form $a(b(x)) \approx c(d(x))$ (no ground equations) is also possible. Note that the word problem in such a system is decidable, since equations of the above type are length-preserving.

6.2 Related Work

Oyamaguchi (1990) shows that the word problem for right-ground TRS is undecidable whereas the word problem in left-linear and right-ground TRS is decidable in polynomial time. In the undecidability proof for right-ground systems rewrite rules which simulate a transition of one configuration to the next contain non-linear variable occurrences at different depth. Note that non-linear variable occurrences at different depth are excluded from semi-linear systems.

Fassbender & Maneth (1996) investigate decidability of E-unification in theories induced by TRS called top-down tree transducers. Syntactic restrictions based on separated function and constructor alphabets are assumed. E-unification in top-down tree transducers with only one function symbol in the alphabet is shown to be decidable. Due to the constructor-based restrictions the results are difficult to compare to semi-linear theories. Otto, Narendran & Dougherty (1995) show that E-unification is decidable in equational theories axiomatized by monadic, confluent string-rewriting systems.

Kaji et al. (1997) show the recognizability of the right-closure of a certain class of right-linear, confluent TRS applied to a linear term. The variables occurring both in the left and right hand side of a rule $l \to r$ are assumed to be linear in l and, moreover, l and the subterms of r are related by additional restrictions which can be effectively computed. The techniques presented in Subsection 3.1 provide a decision method for some restricted unifiability problems modulo the above systems. Actually, the problem addressed in (Kaji et al. 1997) is more general because they deal with "constrained substitutions" which range in some recursively defined (recognizable) set of terms.

Comon et al. (1994) investigate the properties of non-linear shallow theories which are an instance of semi-linear equational theories. Shallow presentations can be transformed into equivalent cycle-syntactic presentations for which decidability of unification has been shown. The first-order theory of the quotient algebra $T(F)/_{=_E}$ is also shown to be decidable where F is finite and E is shallow. However, the proof techniques are entirely different to our approach.

Nieuwenhuis (1996) generalizes the result of Comon et al. (1994) to so-called standard theories. Standard theories extend non-linear shallow theories in a way that non-ground terms containing linear (non-significant) variables are allowed in certain restricted positions in both sides of the equations. An equation $f(s_1, \ldots, s_n) = g(t_1, \ldots, t_m)$ may contain linear terms s_i, respectively t_i, where all other equations with top symbol f, respectively g, must have linear terms in position i. Non-linear variable occurrences are limited to shallow positions [4]. The saturation-based methods are closely related to our work. The decidability results are also obtained by termination

[4]Another extension included in standard theories is discussed in Section 5.

analyses of saturation under basic superposition.

Limet & Réty (1997) show the decidability of E-unification in theories represented by a particular class of confluent, constructor-based TRS. The set of possibly infinite solutions is represented by Tree Tuple Synchronized Grammars. A TRS is transformed into such a grammar which then simulates narrowing. The additional restrictions on the TRS are purely syntactic. However, semi-linear systems are difficult to compare to the constructor-based systems in this approach.

7 Conclusions and Future Work

We have shown that unifiability modulo a sorted shallow equational theory is decidable by means of saturation methods under sorted superposition. With the help of a transformation procedure this result extends to (sorted) semi-linear equational theories. Our result strictly extends previous work concerning shallow theories by Comon et al. (1994). It can be obviously extended into sorted equational theories and also into a generalization of Nieuwenhuis (1996). However, we currently do not have any complexity results concerning the decision procedure or the number of generated mgus. The presented theory is already included in the first-order theorem prover SPASS (Weidenbach 1997, Weidenbach et al. 1998) that can be used for experiments with respect to the presented results.

Let us conclude with another possible improvement of this work. Sorted shallow equational theories generalize $Rec_{=}$ tree automata. To subsume the whole class $Rec_{\neq}$ (Bogaert & Tison 1992), it is necessary to add syntactic disequations to clauses while preserving decidability results concerning membership and emptiness problems. This may have interesting applications in call-by-need normalization strategies for TRS. Durand & Middeldorp (1997) use tree automata techniques both to apply a call-by-need strategy based on the detection of needed redexes and to characterize the class of rewrite systems for which it is effective. The key idea is, given a rewrite system $\mathcal{R}$, to recognize the closure $(\xrightarrow[\mathcal{S}]{*})(\mathrm{NF}_{\mathcal{S}})$ by $\mathcal{S}$ of the set of ground $\mathcal{S}$-normal-forms, where $\mathcal{S}$ is a certain approximation of $\mathcal{R}$. If we approximate $\mathcal{R}$ into a non-linear shallow system $\mathcal{S}$, the above set could be a recognized sorted shallow equational theory with syntactic disequalities, generalizing $Rec_{\neq}$ automata. Thus, with the appropriate extension of the theory of needed-redexes, more general call-by-need normalization strategies for some classes of non-linear rewrite systems could be obtained.

References

Bachmair, L., Ganzinger, H., Lynch, C. & Snyder, W. (1995), 'Basic paramodulation', *Information and Computation* **121**(2), 172–192.

Bogaert, B. & Tison, S. (1992), Equality and disequality constraints on direct subterms in tree automata, *in* A. Finkel & M. Jantzen, eds, 'Proceedings of 9th Annual Symposium on Theoretical Aspects of Computer Science, STACS92', Vol. 577 of *LNCS*, Springer, pp. 161–171.

Comon, H. (1995), Sequentiality, second order monadic logic and tree automata, *in* 'Proceedings 10th IEEE Symposium on Logic in Computer Science, LICS'95', IEEE Computer Society Press, pp. 508–517.

Comon, H., Haberstrau, M. & Jouannaud, J.-P. (1994), 'Syntacticness, cycle-syntacticness and shallow theories', *Information and Computation* **111**(1), 154 –191.

Dershowitz, N. & Jouannaud, J.-P. (1990), Rewrite systems, *in* J. van Leeuwen, ed., 'Handbook of Theoretical Computer Science', Vol. B, Elsevier Science Publishers, chapter 6, pp. 243–320.

Durand, I. & Middeldorp, A. (1997), Decidable call by need computations in term rewriting (extended abstract), *in* W. McCune, ed., 'Automated Deduction – CADE-14, 14th International Conference on Automated Deduction', LNCS 1249, Springer-Verlag, Townsville, North Queensland, Australia, pp. 4–18.

Fassbender, H. & Maneth, S. (1996), A strict border for the decidability of E-unification for recursive functions, *in* M. Hanus & M. Rodríguez-Artalejo, eds, 'Algebraic and Logic Programming (ALP-5) : 5th international conference, Aachen, Germany, September 25-27, 1996', Vol. 1139 of *Lecture notes in computer science*, Springer, Berlin.

Frühwirth, T., Shapiro, E., Vardi, M. & Yardeni, E. (1991), Logic programs as types of logic programs, *in* 'Proc. 6th Symposium on Logics in Computer Science (LICS)', pp. 300–309.

Ganzinger, H., Meyer, C. & Weidenbach, C. (1997), Soft typing for ordered resolution, *in* 'Proceedings of the 14th International Conference on Automated Deduction, CADE-14', Vol. 1249 of *LNAI*, Springer, Townsville, Australia, pp. 321–335.

Jacquemard, F. (1996), Decidable approximations of term rewriting systems, *in* H. Ganzinger, ed., 'Rewriting Techniques and Applications, 7th International Conference, RTA-96', Vol. 1103 of *LNCS*, Springer, pp. 362–376.

Jacquemard, F., Meyer, C. & Weidenbach, C. (1998), Unification in extensions of shallow equational theories, *in* T. Nipkow, ed., 'Rewriting Techniques and Applications, 9th International Conference, RTA-98', Vol. 1379 of *LNCS*, Springer, pp. 76–90.

Jouannaud, J.-P. & Kirchner, C. (1991), Solving equations in abstract algebras: A rule-based survey of unification, *in* J. Lassez & G. Plotkin, eds, 'Computational Logic, Essays in Honor of Alan Robinson', MIT Press, chapter 8, pp. 257–321.

Joyner Jr., W. H. (1976), 'Resolution strategies as decision procedures', *Journal of the ACM* **23**(3), 398–417.

Kaji, Y., Toru, F. & Kasami, T. (1997), 'Solving a unification problem under constrained substitutions using tree automata', *Journal of Symbolic Computation* **23**, 79–117.

Kirchner, C. (1986), Computing unification algorithms, *in* 'Proceedings of the First Symposium on Logic in Computer Science', Cambridge, Massachusetts, pp. 206–216.

Kozen, D. (1981), 'Positive first order logic is NP-complete.', *IBM Journal of Res. Develop.* **25**(4), 327–332.

Limet, S. & Réty, P. (1997), E-unification by means of tree tuple synchronized grammars, *in* M. Bidoit & M. Dauchet, eds, 'TAPSOFT '97: Proceedings of the Seventh Joint Conference on Theory and Practice of Software Development, 7th International Joint Conference CAAP/FASE, Lille, France', Vol. 1214, Springer-Verlag, Lille, France.

Nieuwenhuis, R. (1996), Basic paramodulation and decidable theories (extended abstract), *in* 'Proceedings 11th IEEE Symposium on Logic in Computer Science, LICS'96', IEEE Computer Society Press, pp. 473–482.

Nieuwenhuis, R. & Rubio, A. (1995), 'Theorem proving with ordering and equality constrained clauses', *Journal of Symbolic Computation* **19**, 321–351.

Otto, F., Narendran, P. & Dougherty, D. J. (1995), Some independence results for equational unification, *in* J. Hsiang, ed., 'Rewriting Techniques and Applications, 6th International Conference, RTA-95', LNCS 914, Springer-Verlag, Kaiserslautern, Germany, pp. 367–381.

Oyamaguchi, M. (1990), 'On the word problem for right-ground term-rewriting systems', *The Transactions of the IEICE* **73**(5), 718–723.

Seidl, H. (1994), 'Haskell overloading is DEXPTIME-complete', *Information Processing Letters* **52**(2), 57–60.

Weidenbach, C. (1996), Computational Aspects of a First-Order Logic with Sorts, Dissertation, Technische Fakultät der Universität des Saarlandes, Saarbrücken, Germany.

Weidenbach, C. (1997), 'Spass version 0.49', *Journal of Automated Reasoning* **18**(2), 247–252.

Weidenbach, C. (1998), Sorted unification and tree automata, *in* W. Bibel & P. H. Schmitt, eds, 'Automated Deduction - A Basis for Applications', Vol. 1 of *Applied Logic*, Kluwer, chapter 9, pp. 291–320.

Weidenbach, C., Meyer, C., Cohrs, C., Engel, T. & Keen, E. (1998), 'Spass v0.77', *Journal of Automated Reasoning* **21**(1), 113–113.

I N F O R M A T I K

Below you find a list of the most recent technical reports of the Max-Planck-Institut für Informatik. They are available by anonymous ftp from `ftp.mpi-sb.mpg.de` under the directory `pub/papers/reports`. Most of the reports are also accessible via WWW using the URL `http://www.mpi-sb.mpg.de`. If you have any questions concerning ftp or WWW access, please contact `reports@mpi-sb.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. Birgit Hofmann
Im Stadtwald
D-66123 Saarbrücken
GERMANY
e-mail: `library@mpi-sb.mpg.de`

MPI-I-98-2-018	F. Eisenbrand	A Note on the Membership Problem for the First Elementary Closure of a Polyhedron
MPI-I-98-2-017	M. Tzakova, P. Blackburn	Hybridizing Concept Languages
MPI-I-98-2-014	Y. Gurevich, M. Veanes	Partisan Corroboration, and Shifted Pairing
MPI-I-98-2-013	H. Ganzinger, F. Jacquemard, M. Veanes	Rigid Reachability
MPI-I-98-2-012	G. Delzanno, A. Podelski	Model Checking Infinite-state Systems in CLP
MPI-I-98-2-011	A. Degtyarev, A. Voronkov	Equality Reasoning in Sequent-Based Calculi
MPI-I-98-2-010	S. Ramangalahy	Strategies for Conformance Testing
MPI-I-98-2-009	S. Vorobyov	The Undecidability of the First-Order Theories of One Step Rewriting in Linear Canonical Systems
MPI-I-98-2-008	S. Vorobyov	AE-Equational theory of context unification is Co-RE-Hard
MPI-I-98-2-007	S. Vorobyov	The Most Nonelementary Theory (A Direct Lower Bound Proof)
MPI-I-98-2-006	P. Blackburn, M. Tzakova	Hybrid Languages and Temporal Logic
MPI-I-98-2-005	M. Veanes	The Relation Between Second-Order Unification and Simultaneous Rigid E-Unification
MPI-I-98-2-004	S. Vorobyov	Satisfiability of Functional+Record Subtype Constraints is NP-Hard
MPI-I-98-2-003	R.A. Schmidt	E-Unification for Subsystems of S4
MPI-I-98-1-031	G.W. Klau, P. Mutzel	Optimal Compaction of Orthogonal Grid Drawings
MPI-I-98-1-030	H. Brönniman, L. Kettner, S. Schirra, R. Veltkamp	Applications of the Generic Programming Paradigm in the Design of CGAL
MPI-I-98-1-029	P. Mutzel, R. Weiskircher	Optimizing Over All Combinatorial Embeddings of a Planar Graph
MPI-I-98-1-028	A. Crauser, K. Mehlhorn, E. Althaus, K. Brengel, T. Buchheit, J. Keller, H. Krone, O. Lambert, R. Schulte, S. Thiel, M. Westphal, R. Wirth	On the performance of LEDA-SM
MPI-I-98-1-027	C. Burnikel	Delaunay Graphs by Divide and Conquer
MPI-I-98-1-026	K. Jansen, L. Porkolab	Improved Approximation Schemes for Scheduling Unrelated Parallel Machines
MPI-I-98-1-025	K. Jansen, L. Porkolab	Linear-time Approximation Schemes for Scheduling Malleable Parallel Tasks
MPI-I-98-1-024	S. Burkhardt, A. Crauser, P. Ferragina, H. Lenhof, E. Rivals, M. Vingron	q-gram Based Database Searching Using a Suffix Array (QUASAR)

MPI-I-98-1-023	C. Burnikel	Rational Points on Circles
MPI-I-98-1-022	C. Burnikel, J. Ziegler	Fast Recursive Division
MPI-I-98-1-021	S. Albers, G. Schmidt	Scheduling with Unexpected Machine Breakdowns
MPI-I-98-1-020	C. Rüb	On Wallace's Method for the Generation of Normal Variates
MPI-I-98-1-019		2nd Workshop on Algorithm Engineering WAE '98 - Proceedings
MPI-I-98-1-018	D. Dubhashi, D. Ranjan	On Positive Influence and Negative Dependence
MPI-I-98-1-017	A. Crauser, P. Ferragina, K. Mehlhorn, U. Meyer, E. Ramos	Randomized External-Memory Algorithms for Some Geometric Problems
MPI-I-98-1-016	P. Krysta, K. Loryś	New Approximation Algorithms for the Achromatic Number
MPI-I-98-1-015	M.R. Henzinger, S. Leonardi	Scheduling Multicasts on Unit-Capacity Trees and Meshes
MPI-I-98-1-014	U. Meyer, J.F. Sibeyn	Time-Independent Gossiping on Full-Port Tori
MPI-I-98-1-013	G.W. Klau, P. Mutzel	Quasi-Orthogonal Drawing of Planar Graphs
MPI-I-98-1-012	S. Mahajan, E.A. Ramos, K.V. Subrahmanyam	Solving some discrepancy problems in NC*
MPI-I-98-1-011	G.N. Frederickson, R. Solis-Oba	Robustness analysis in combinatorial optimization
MPI-I-98-1-010	R. Solis-Oba	2-Approximation algorithm for finding a spanning tree with maximum number of leaves
MPI-I-98-1-009	D. Frigioni, A. Marchetti-Spaccamela, U. Nanni	Fully dynamic shortest paths and negative cycle detection on diagraphs with Arbitrary Arc Weights
MPI-I-98-1-008	M. Jünger, S. Leipert, P. Mutzel	A Note on Computing a Maximal Planar Subgraph using PQ-Trees
MPI-I-98-1-007	A. Fabri, G. Giezeman, L. Kettner, S. Schirra, S. Schönherr	On the Design of CGAL, the Computational Geometry Algorithms Library
MPI-I-98-1-006	K. Jansen	A new characterization for parity graphs and a coloring problem with costs
MPI-I-98-1-005	K. Jansen	The mutual exclusion scheduling problem for permutation and comparability graphs
MPI-I-98-1-004	S. Schirra	Robustness and Precision Issues in Geometric Computation
MPI-I-98-1-003	S. Schirra	Parameterized Implementations of Classical Planar Convex Hull Algorithms and Extreme Point Compuations
MPI-I-98-1-002	G.S. Brodal, M.C. Pinotti	Comparator Networks for Binary Heap Construction
MPI-I-98-1-001	T. Hagerup	Simpler and Faster Static AC^0 Dictionaries
MPI-I-97-2-012	L. Bachmair, H. Ganzinger, A. Voronkov	Elimination of Equality via Transformation with Ordering Constraints
MPI-I-97-2-011	L. Bachmair, H. Ganzinger	Strict Basic Superposition and Chaining
MPI-I-97-2-010	S. Vorobyov, A. Voronkov	Complexity of Nonrecursive Logic Programs with Complex Values
MPI-I-97-2-009	A. Bockmayr, F. Eisenbrand	On the Chvátal Rank of Polytopes in the 0/1 Cube
MPI-I-97-2-008	A. Bockmayr, T. Kasper	A Unifying Framework for Integer and Finite Domain Constraint Programming
MPI-I-97-2-007	P. Blackburn, M. Tzakova	Two Hybrid Logics
MPI-I-97-2-006	S. Vorobyov	Third-order matching in $\lambda \rightarrow$-Curry is undecidable
MPI-I-97-2-005	L. Bachmair, H. Ganzinger	A Theory of Resolution
MPI-I-97-2-004	W. Charatonik, A. Podelski	Solving set constraints for greatest models
MPI-I-97-2-003	U. Hustadt, R.A. Schmidt	On evaluating decision procedures for modal logic