Decidable Matching for Convergent Systems*

Nachum Dershowitz, Subrata Mitra
Department of Computer Science
University of Illinois
Urbana, IL 61801, U.S.A.
{nachum, mitra}@cs.uiuc.edu

G. Sivakumar
Department of Computer Science
Indian Institute of Technology
Bombay, 400076, India
siva@cse.iitb.ernet.in

Abstract

We describe decision procedures for certain classes of semantic matching problems, where the equational theory with respect to which the semantic matching is performed has a convergent rewrite system. We give counterexamples to show that semantic matching becomes undecidable (as it generally is) when the conditions we give are weakened.

1 Introduction

Equation solving is the process of finding a substitution which makes two terms equal in a given theory, while *semantic unification* is the process that generates a basis set of such unifiers. A simpler version of this problem, *semantic matching*, restricts the substitution to apply only to the term on the left, say (the *pattern*). Semantic matching has potential applications in pattern-directed languages. For example, if we could match with respect to addition, the function definition

$$\begin{aligned} half(x+x) &= x \\ half(x+x+1) &= x \end{aligned}$$

could be applied to a term like $half(17)$, by finding that the pattern in the second definition matches the term when $x = 8$.

It is well-known that any strategy for finding a complete set of unifiers (or matchings) for two terms, with respect to a given theory, may not terminate, even when the theory is presented as a finite and *convergent* (terminating and confluent) set of rewrite rules [HH87, Bo87]. But, for some special classes of theories—associativity and commutativity, for instance—semantic unification is decidable.

It is, therefore, of interest to find suitable cases for which a particular equation-solving procedure is provably terminating, thus implying that the semantic unification or semantic matching problems in the corresponding theories are decidable. In this paper, we consider only equational theories for which there is a finite convergent rewrite system. We specialize the unification procedure given in [DS87, Mit90, JK91] and study the effect of some syntactic and semantic restrictions on the rewrite system presenting a theory, which result in decidability.

In the remainder of this section, we briefly review the relevant basic notions, terminology and results for equational theories and rewrite systems. For surveys of this area, see [DJ90] and [JK91].

Terms are constructed from a given set of function symbols and variables. We normally use l, r, s, and t for arbitrary terms, and x, y, and z for variables. A *ground* term is one containing

*This research was supported in part by the U. S. National Science Foundation under Grants CCR-90-07195 and CCR-90-24271.

no variables (such as, $0+0$). A term t is said to be *linear* in a variable x if x occurs only once in t. For example, the term $x + s(y) * z$ is linear in all three variables. The *size* of a ground term is the number of function symbols it has, whereas its *depth* is the length of the longest path in its tree representation. A *substitution* is a mapping from variables to terms. We use lower case Greek letters θ, σ and μ to denote substitutions, and write them out as $\{x_1 \mapsto s_1, \ldots, x_m \mapsto s_m\}$.

A (ground) term t *matches* a pattern (term) s in a theory E if $E \models s\sigma = t$ for some substitution σ. We also say that t is an *instance* of s in this case. For example, $0 + s(0)$ matches $y + x$ with the substitution $\{x \mapsto 0, y \mapsto s(0)\}$ in the theory $\{x + y = y + x\}$. A term s *unifies* with a term t in a theory E if $E \models s\sigma = t\sigma$ for some substitution σ. We say that a solution σ is *at least as general* as a solution ρ if there exists a substitution τ such that ρ and the composition of σ and τ give equal terms (equal, in E), for each variable in the problem. For example, a most general unifier of $x + y$ and $u + v$ is the substitution $\{x \mapsto u, y \mapsto v\}$. Semantic unification is the process of finding all such substitutions.

An *equation* is an *unordered* pair of terms, written in the form $s = t$. Either or both of s and t may contain variables; which are understood as being universally quantified. A *rewrite rule* is an oriented equation between terms, written $l \rightarrow r$; a *rewrite system* is a set of such rules. A rewrite rule is *left linear* if its left hand side is linear for all the variables, for example $s(x) * y \rightarrow y + (x * y)$. A rewrite rule is said to be *variable-preserving* if all the variables in its left hand side also appear in its right hand side term. A function symbol f is said to be a *defined function* with respect to a rewrite system R if there exists a rule in R with f as the top-most symbol of its left hand side; if there is no such rule, then f is called a *constructor*. We will use $\equiv$ for identity of terms, to distinguish it from other forms of equality.

For a given system R, the rewrite relation $\rightarrow$ replaces an instance $l\sigma$ of a left-hand side l by the corresponding instance $r\sigma$ of the right-hand side r. Unlike equations, replacements are not allowed in the reverse direction. We write $s \rightarrow t$, if s rewrites to t in one step; $s \rightarrow^* t$, if t is *derivable* from s, that is, if s rewrites to t in zero or more steps; $s \downarrow t$, if s and t *join* , that is, if $s \rightarrow^* w$ and $t \rightarrow^* w$ for some term w. A term s is said to be *irreducible*, or in *normal form*, if there is no term t such that $s \rightarrow t$. We write $s \rightarrow^! t$ if $s \rightarrow^* t$ and t is in normal form. All the matching problems we consider are of the form $s \rightarrow^? N$, meaning: find a substitution σ such that $s\sigma$ has normal form N. (We will frequently use N to stand for a term in normal form.) A solution is *irreducible* if each of the terms substituted for the variables in the equation is irreducible.

A rewrite relation is *terminating* if there is no infinite chain of rewrites: $t_1 \rightarrow t_2 \rightarrow \cdots \rightarrow t_k \rightarrow \cdots$. A rewrite relation is *ground confluent* if whenever two ground terms, s and t, are derivable from a term u, then a term v is derivable from both s and t. That is, if $u \rightarrow^* s$ and $u \rightarrow^* t$, then $s \rightarrow^* v$ and $t \rightarrow^* v$ for some term v. A rewrite system that is both ground confluent and terminating is said to be *ground convergent*; whenever we say "convergent" in this paper, we mean "ground convergent". Convergent rewrite systems are important for the following reason: If R is a convergent rewrite system and E is the underlying equational theory (E is R with each rule taken as an equation), then $E \models s = t$ (for ground terms s and t) iff $s \downarrow t$ in R.

Example 1.1. *The following convergent system has an undecidable semantic unification problem.*

$$\begin{aligned} 1 + x &\rightarrow s(x) \\ s(x) + y &\rightarrow s(x + y) \end{aligned}$$

$$\begin{aligned} 1 * x &\rightarrow x \\ s(x) * 1 &\rightarrow s(x) \\ s(x) * s(y) &\rightarrow s(y + (x * s(y))) \end{aligned}$$

The system defines addition ($+$) and multiplication ($*$) over positive integers, which are represented in unary notation, using the constant 1 and successor function s.

It can be shown that in general it is undecidable if an equation of has a solution with respect to the rewrite system given above, since were there a decision procedure for this, then it would solve Hilbert's undecidable Tenth Problem. We will prove later that the semantic matching problem is, nevertheless, decidable for this theory. ([Bo87, DJ90] use similar examples to show that in general semantic matching and unification are undecidable even for convergent systems.)

In the most general case, however, semantic matching can be as difficult as full semantic unification: For example, adding a new rule $eq(x, x) \rightarrow true$ to the above example makes unifying two terms s and t the same as matching $eq(s, t)$ to $true$ in the augmented theory.

2 The Matching Procedure

We describe a method for semantic matching that is complete for the special cases of matchings that we will consider in Section 3, and later in Section 4. This is a simplified version of the generally complete system for unification appearing in [DS87, Mit90, JK91], which is a refinement of *narrowing*, as studied in [**?**, Hul80, NRS89, Ret87], and others.

We consider equational theories that are given as finite convergent rewrite systems. Convergent systems allow one to ignore reducible solutions to semantic unification and matching problems. For an equational goal like $s(0) + x \rightarrow^? s(s(0))$, in the theory of addition ($\{0 + x = x, s(x) + y = s(x + y)\}$), the only solution of interest is $\{x \mapsto s(0)\}$. Reducible solutions, like $\{x \mapsto 0 + s(0)\}$, are redundant if we collect all irreducible ones. We will, therefore, be interested only in finding solutions at least as general as all the irreducible ones. In the decidable cases we describe, there are only finitely many such solutions.

We always begin with a goal of the form $s \rightarrow^? N$, where N is a ground normal form, since instead of matching s with an arbitrary t, we can take N to be its normal form. The transformation rules keep track of the current list of subgoals to be solved. A matching is found when all the subgoals are of the form $x \mapsto N$, where x is a variable and N is a normal form, provided that whenever the same variable appears on the left in more than one subgoal, the identical term appears on the right. To get a complete set of solutions we need to consider different ways of applying the following (non-deterministic) transformation rules:

Eliminate	$\{x \rightarrow^? t\} \cup G$ $\rightsquigarrow$ $\{x \mapsto t\} \cup G\{x \mapsto t\}$

Decompose	$\{f(s_1, \ldots, s_n) \rightarrow^? f(t_1, \ldots, t_n)\} \cup G$ $\rightsquigarrow$ $\{s_1 \rightarrow^? t_1, \ldots, s_n \rightarrow^? t_n\} \cup G$

Mutate	$\{f(s_1, \ldots, s_n) \rightarrow^? t\} \cup G$ $\rightsquigarrow$ $\{s_1 \rightarrow^? l_1, \ldots, s_n \rightarrow^? l_n, r \rightarrow^? t\} \cup G$ where $f(l_1, \ldots, l_n) \rightarrow r$ is a renamed rule in R

We need not try all transformations on all goals, as shown in the proof of the following completeness result:

Theorem 2.1 (Completeness). *Let R be a variable-preserving convergent rewrite system. If the*

goal $s \rightarrow^? N$ *has a solution* θ *(that is,* $s\theta \rightarrow^! N$*), then there is a derivation of the form*

$$\{s \stackrel{?}{\rightarrow} N\} \stackrel{!}{\rightsquigarrow} \mu,$$

such that μ *(viewed as a substitution) is at least as general as* θ*.*

Proof. The first observation is that if $s \equiv f(s_1, \ldots, s_n)$, and we consider an innermost rewriting strategy, then

$$f(s_1\theta, \ldots, s_n\theta) \rightarrow^* f(N_1, \ldots, N_n),$$

where $s_i\theta \rightarrow^! N_i$. (If s is a variable, then $s \mapsto N$ is the solution we're looking for.) Thus, at this stage there are two possibilities for the topmost position of the right hand side:

- No rule applies at this position, and thus $N \equiv f(N_1, \ldots, N_n)$. This situation is simulated by the **Decompose** transformation rule, which generates the subgoals corresponding to $s_i\theta \rightarrow^! N_i$.
- Some rule applies at this position. This is handled by the **Mutate** transformation rule.

After a finite number of decompositions, the mutation corresponding to the next rewrite step in the derivation of N from $s\theta$ becomes possible, making progress towards the desired solution.

We show next that, since R is variable-preserving, we need only deal with subgoals which have ground normal forms on the right. This guarantees that whenever we have a subgoal with a variable on the left, no further work remains. Clearly, **Decompose** preserves this property of right hand sides. **Mutate** does not, since the l_i may have variables in them. But, if we solve $r \rightarrow^? t$ first, then we can apply (using **Eliminate**) the solutions we get to *each* of the variables in the l_i. (We get ground substitutions for all of them, on account of the system's being variable-preserving.) Since we need only look at innermost rewriting, the $l_i\sigma$ must be in normal form.

Finally, we have to show that the computed answer (μ) is at least as general as the solution θ. This can be done by induction on the well-founded multiset extension of $\rightarrow^+$, which compares multisets of the left hand side terms of a list of goals, with the solution θ applied to each such term, along any suitable derivation sequence. □

We can similarly show completeness for the case when the rewrite system is left linear.

The termination proofs in later sections assume particular strategies for selecting subgoals or discarding subgoals which are instances of the selection strategy used in the above completeness proof, namely, always find solutions to the last subgoal of **Mutate** first, and eliminate goals whenever possible. Of course, **Decompose** and **Mutate** may both apply to the same subgoal, and there may be many ways of mutating, one for each rule of the rewrite system with the same outermost symbol as the left side of the subgoal.

3 Variable-Preserving Rules

In looking for decidable matching problems, we started with the following result (a special case of Theorem 3.6 which we prove later): *If* R *is a variable-preserving convergent term rewriting system for which:*

- *all right hand sides of rules are either variables, or have a constructor at the top-level, and*
- *there are no nested defined functions in any right hand side,*

then the semantic matching problem is decidable for R*.*

Example 3.1. *By this result, the following system has a decidable semantic matching problem.*

$$\begin{array}{rcl} app(nil, x) & \rightarrow & x \\ app(cons(a, x), y) & \rightarrow & cons(a, app(x, y)) \end{array}$$

In [HH87], there is an example of a system with a single defined function in every right hand side, which has an undecidable semantic matching problem. There, the defined function on the right hand side of rules does not appear below a constructor, but it obeys the other restrictions. This shows that defined functions must appear below at least one constructor.

Next, we tried to allow some nested function symbols on the right hand side of the rewrite rules. We require the following definitions:

Definition 3.2 (Suitable Property). A *suitable property* is a measure (like *depth*, *size*, etc.) associated with ground terms, along with a well-founded total ordering $>$ which compares values of P, such that P is strictly larger, under $>$, for terms than for its subterms.

Definition 3.3 (Non-Decreasing). Let P be a suitable property. A function symbol f is defined to be *non-decreasing* (with respect to P) if whenever $f(\widehat{s_1}, \ldots, \widehat{s_n}) \rightarrow^! N$, where each $\widehat{s_i}$ is in ground normal form, $P(\widehat{s_i}) \leq P(N)$. Any function which does not have this property is said to be a *potentially decreasing* function (with respect to P).

We can similarly define the notion of strict increasingness of a function.

It is not possible to always decide whether a function defined by a given convergent rewrite system is non-decreasing with respect to a property P, even for a simple suitable property like depth.

Lemma 3.4. *It is undecidable if a function symbol is depth non-decreasing.*

Proof. Consider

$$\begin{array}{rcl} g(x) & \rightarrow & h(f(S_1 \circ \$, S_2 \circ \$, x), x) \\ h(f(\$, \$, \$), x) & \rightarrow & \$ \end{array}$$

where f is as detailed in [HH87]. If S_1 and S_2 are respectively the start symbols for two context free grammars G_1 and G_2, we have

$$f(S_1 \circ \$, S_2 \circ \$, x) \rightarrow^! f(\$, \$, \$) \textit{ iff } x \in G_1 \textit{ and } x \in G_2.$$

By the above construction, g can be depth non-decreasing if and only if

$$\forall x . x \notin (G_1 \cap G_2).$$

Thus, a decision procedure for this problem could be used to decide if the intersection of two arbitrary context free grammars is empty, which is impossible. □

This lemma shows that in general it is not possible to decide if a function is depth (increasing) non-decreasing, even for convergent systems. However, certain decidable subclasses with the property are easy to identify. For example, any function which has a variable dropping rule cannot be depth (increasing) non-decreasing. Again, for each rule $l \rightarrow r$ which defines a function f, if $depth(l) \leq depth(r)$ then f is depth non-decreasing. We can also have similar sufficient conditions using the depth of each variable in the rule. For example, if every variable occurs below at least the same number of constructors on the right side, as on the left, then the corresponding function is depth non-decreasing. We can use the last criterion to show that $+$, as defined in Example 1.1, is depth non-decreasing.

Unfortunately, if the right hand sides in rewrite rules have defined functions nested below a potentially (depth) decreasing function, then the resulting system may have undecidable semantic matching problems. This we show by considering the rules shown below, together with the definitions of $+$ and $*$ given in Example 1.1.

Example 3.5.

$$\begin{array}{rcl} \mathit{half}(s(1)) & \rightarrow & 1 \\ \mathit{half}(s(s(x)) & \rightarrow & s(\mathit{half}(x)) \end{array}$$

$$\begin{array}{rcl} f(1,1) & \rightarrow & s(1) \\ f(s(x),s(y)) & \rightarrow & s(\mathit{half}(f(x,y))) \end{array}$$

Here *half* is a potentially (depth) decreasing function. We have the following property for f:

$$f(x,y) = \begin{cases} s(1) & \text{if } x = y \\ \mathit{undefined} & \text{otherwise} \end{cases}$$

Thus, we can now try to solve the goal $f(t_1,t_2) \rightarrow^? s(1)$, where t_1 and t_2 are terms involving $+$ and $*$. This goal has a solution σ iff $t_1\sigma$ and $t_2\sigma$ have the same ground normal form (because of the observation about f). Thus, if this problem has a decision procedure, then we could use the same for deciding the semantic unification problem mentioned in Example 1.1. Therefore, no such decision procedure can exist.

Based on the counterexample above, it can be seen that a function definition in terms of some potentially decreasing functions is not suitable for our purpose, and we therefore restrict the right hand sides of rules to have potentially decreasing functions only at the lowest level (that is, no other defined function symbol can be nested below them). We have:

Theorem 3.6. *Let R be a convergent variable-preserving term rewriting system, and P be some suitable property. If*

- *all right hand sides for rules in R are either variables, or have a constructor at the top-level, and*
- *all right hand sides are such that no defined function is nested below any function decreasing with respect to P,*

then the semantic matching problem is decidable for R.

Proof. Let $\succ$ be a well-founded ordering on goals such that $s_1 \rightarrow^? N_1 \succ s_2 \rightarrow^? N_2$ iff either $P(N_1) > P(N_2)$ or $P(N_1) = P(N_2)$ and s_2 is a subterm of s_1.

We prove that it is possible to find all solutions (in finite time) to any goal of the form $\varrho \rightarrow^? N$, where ϱ is a term which has no defined function nested below any decreasing function and N is a ground normal form. This we do by induction with respect to the ordering $\succ$.

The interesting case is the one in which $\varrho \equiv f(\varrho_1,\ldots,\varrho_n)$, and f is a defined function. It is therefore possible to use the transformation rule **Mutate** on this goal, applying some rule $f(l_1,\ldots,l_n) \rightarrow \rho$. The essential steps are:

$$\begin{array}{lll} \{\varrho \rightarrow^? N\} & \rightsquigarrow_{\mathbf{Mutate}} & \{\varrho_i \rightarrow^? l_i, \rho \rightarrow^? N\}, 1 \le i \le n \\ & \rightsquigarrow_{\mathbf{Decompose}} & \{\varrho_i \rightarrow^? l_i, \rho_j \rightarrow^? N_j\} \end{array}$$

Since every right hand side, by assumption, has constructors at the top, we have shown the decomposition step which may be applied to $\rho \rightarrow^? N$.

The subgoals $\{\rho_j \rightarrow^? N_j\}$ produced after the decomposition step are smaller than the original goal, that is, $\{\varrho \rightarrow^? N\} \succ \{\rho_j \rightarrow^? N_j\}$, for each j. Thus, by applying the inductive hypothesis we can assume that all the solutions to each of the goals in $\{\rho_j \rightarrow^? N_j\}$ (and therefore also for their collection, that is, $\rho \rightarrow^? N$) can be found in finite time. Let σ be the solution obtained along one

feasible branch for the goal $\rho \rightarrow^? N$. Now, since all the rules are variable-preserving, ρ contains all the variables that are in any of the l_i terms. Furthermore, because of the variable-preserving nature of all rules, any such σ must be a ground solution (if not, we will have a situation where a non-ground term will rewrite using only variable-preserving rules to a ground term, which is not possible). Thus, for any such σ, each $l_i\sigma$ term must be ground.

There are now two different cases to be considered.

- Function f is *potentially decreasing*. By assumption there is no defined function below it, that is, no ϱ_i has a defined function, and therefore all the $\varrho_i \rightarrow^? l_i$ subgoals can be decomposed immediately to solved forms ($x \mapsto N'$), or to unsolvable goals with different constructors at the top. This shows that all the solutions for $\varrho \rightarrow^? N$ can be found in finite time in this case.

- Function f is *non-decreasing*. The important point to note is that each left hand side in the list of subgoals has the property that no defined function is nested below a potentially decreasing function. Let us now consider the ground solution σ as described above. Since f is known to be non-decreasing with respect to P, each of the $l_i\sigma$ terms must be such that $P(l_i\sigma) \leq P(N)$, or else the partial solution σ is not feasible for f, and can be ignored. (In this case, the goal $\varrho \rightarrow^? N$ has no solution using this rule for **Mutate**.)

  Thus, for all feasible paths, we have that $P(l_i\sigma) \leq P(N)$, and therefore, we get

  $$\{\varrho \stackrel{?}{\rightarrow} N\} \succ \{\varrho_i \stackrel{?}{\rightarrow} l_i\sigma\},$$

  for each i, since each ϱ_i is a subterm of ϱ. Thus, by the induction hypothesis, each subgoal $\varrho_i \rightarrow^? l_i\sigma$ can be solved, and therefore the goal $\varrho \rightarrow^? N$ itself can also be solved.

Using this result, it is easy to show (by induction on the size of the left-sides of goals) that for any term s (even without the restrictions imposed on ϱ), and ground normal form $\widehat{N}$, the goal $s \rightarrow^? \widehat{N}$ is solvable. The idea is that for every application of **Mutate** with the goal $s \rightarrow^? \widehat{N}$, the subgoal $r \rightarrow^? N$ is solvable (by above argument). Thus, we can replace the multiset of subgoals generated by **Mutate**, by a finite number of such multisets (without $r \rightarrow^? N$) corresponding to each of the solutions of $r \rightarrow^? N$. □

A few comments about the restrictions used for the proof are in order:

- Theorem 3.6 uses semantic restrictions on the right hand sides of rewrite rules of the system, by requiring that certain defined functions can appear only below non-decreasing functions. Although there can be no decision procedure to check if a function is increasing in general, certain simple sufficient restrictions are easy to check, given the corresponding set of rewrite rules. For instance, the special case, mentioned at the beginning of this section and applied in Example 3.1, uses only syntactic restrictions.

- The fact that we do not have defined functions nested below potentially decreasing function(s) is important because of the counterexample given in Example 3.5.

In certain special cases it is possible to relax the requirement that all right hand sides with defined functions must have a constructor at the top-level. For example, if we assume that the top-level function on the right hand side is strictly increasing, and that it eventually generates a constructor in a finite number of steps, then the above theorem holds.

The following example illustrates this point:

Example 3.7.

$$\begin{array}{rcl} 1+x & \to & s(x) \\ s(x)+y & \to & s(x+y) \end{array}$$

$$\begin{array}{rcl} fib(1) & \to & 1 \\ fib(s(1)) & \to & 1 \\ fib(s(s(x))) & \to & fib(s(x)) + fib(x) \end{array}$$

Here $+$ is a strictly *(depth) increasing* function, and fib defines the Fibonacci numbers, both being defined over positive integers. Furthermore, both rules for $+$ have constructors at the top-level on the right hand side, and the remaining rules have the properties required by Theorem 3.6. The semantic matching problem is decidable for this system.

The essential idea is that if any sequence of applications of **Mutate** for increasing functions generate a constructor eventually, then the matching problem is decidable. Here is an outline of the proof.

If the top-level symbol (of ϱ in Theorem 3.6) is an increasing function, the applicable transformation rule generates the following derivation.

$$\{\varrho \to^? N\} \quad \rightsquigarrow_{\mathbf{Mutate}} \quad \{\varrho_{i_1} \to^? l_{i_1}, \rho_1 \to^? N\}$$

The goal $\rho \to^? N$ is not decreasing as such. However, since we assumed that all such derivations eventually generate a constructor at the top, we must have at least one step of decomposition if we continue to mutate this goal. The derivation therefore would look like:

$$\begin{array}{lll} \{\varrho \to^? N\} & \rightsquigarrow_{\mathbf{Mutate}^*} & \{\varrho_{i_1} \to^? l_{i_1}, \ldots, \varrho_{i_m} \to^? l_{i_m}, \rho_m \to^? N\} \\ & \rightsquigarrow_{\mathbf{Decompose}^*} & \{\varrho_{i_1} \to^? l_{i_1}, \ldots, \varrho_{i_m} \to^? l_{i_m}, \bar{\rho_m} \to^? \bar{N}\} \end{array}$$

We assume that we only mutate the $\rho_m \to^? N$ subgoal at every stage.

Now, we can show that the subgoals are decreasing with respect to the ordering $\succ$, as in Theorem 3.6.

Also, the system described in Example 1.1 obeys all the restrictions of Theorem 3.6, and thus has a decidable semantic matching problem.

4 Variable Dropping Rules

In this section we deal with the possibility of incorporating variable dropping rules into the rewrite system, and we will try to extend Theorem 3.6 suitably to handle such cases. However, before we do so, we point out some cases which cause problems, by way of counterexamples.

Example 4.1. *Consider the rule given below, with the definition of $+$ and $*$ from Example 1.1*

$$eq(x,x) \quad \to \quad true$$

Here the eq rule is the only one which is variable dropping. For this set of rules, the problem of $\{eq(s,t) \to^? true\}$ is not solvable in general, because once again a solution of this problem would mean a decision procedure for some variation of the Hilbert's Tenth problem.

Example 4.2. *This time we consider two rules and the definitions of $+$ and $*$ as before*

$$\begin{array}{rcl} f(s(x),0) & \to & 0 \\ eq(x,x) & \to & x \end{array}$$

Here the only variable dropping rules is the one for f. Consider a goal of the form $\{f(eq(t_1, t_2), y) \rightarrow^? 0\}$, where t_1 and t_2 are terms involving $+$ and $*$. The possible solution steps for this goal are shown below:

$$\begin{array}{lll} \{f(eq(t_1, t_2), y) \rightarrow^? 0\} & \leadsto_{\mathbf{Mutate}} & \{eq(t_1, t_2) \rightarrow^? s(x_1)\}, \sigma = \{y \mapsto 0\} \\ & \leadsto_{\mathbf{Mutate}} & \{t_1 \rightarrow^? x, t_2 \rightarrow^? x\}, \sigma = \{y \mapsto 0, x \mapsto s(x_1)\} \end{array}$$

Thus, if this goal has a solution, then we can also solve $t_1 \stackrel{?}{=} t_2$, with respect to $+$ and $*$, which is not possible. This example illustrates the fact that a system with a single (left linear) variable dropping rule may admit undecidable matching problems.

The above counterexamples show that a single variable dropping rule may make the goal solving procedure non-terminating. However, it is possible to have variable dropping rules and still have a decidable semantic matching algorithm. We prove the result in Theorem 4.3.

Theorem 4.3. *Let R be a left linear rewrite system and P be a suitable property. If*

- *all right hand sides for rules in R are either variables, or have a constructor at the top-level, and*
- *all right hand sides are such that no defined function is nested below any function decreasing with respect to P,*

then the semantic matching problem is decidable for R.

We need two lemmata.

Lemma 4.4. *Let R be a rewrite system as defined in Theorem 4.3. Then, for the initial goal $\{s \rightarrow^? N\}$, where N is ground, if $G \cup \{t_1 \rightarrow^? t_2\}$ is the list of subgoals generated by the procedure at some point and x is a variable in t_2, then t_2 must be linear with respect to x, and, furthermore, x does not occur in any right hand side in a subgoal in G.*

Proof. If $\tau \rightarrow^? t$ is a subgoal generated by the procedure for the initial goal $s \rightarrow^? N$, then the variables of t must come either from N or from the left hand side of some rule in R. For our case, N is ground, and R is left linear, thus this variable can not occur in any other right hand side in the goal list, and again t itself must be linear in this variable. □

Lemma 4.5. *Let R be a rewrite system as defined in Theorem 4.3. Then, in solving $\{s \rightarrow^? N\}$ a subgoal of the form $s \rightarrow^? x$ can be ignored if x is one of the dropped variables in some rule used in the transformation rule* **Mutate**.

Proof. Let G denote the remaining list of subgoals when $s \rightarrow^? x$ is encountered. By Lemma 4.4, x can not appear in any right hand side in G. Furthermore, since x is a dropped variable (and since every mutation renames variables), x can not be present in any of the left hand sides either or in the partial answer substitution also. Thus, any such goal is trivially solvable (that is, it has a solution for any substitution for the variables in s), and can be ignored. □

We can now state the proof of the theorem.

Proof. In the proof of Theorem 3.6 we never needed to solve for any subgoal of the form $s \rightarrow^? t$, where t was non-ground. This is because of the variable-preserving nature of all the rules: after applying **Mutate** we could first solve the $r \rightarrow^? N$ subgoal to get ground answers for all the variables introduced by any rule, and the application of this ground substitution would make all the corresponding left hand sides of the rule ground.

Since we now have variable dropping rules, this assumption is no longer true. However, due to Lemma 4.5 we only need to solve for goals of the form $s \rightarrow^? t$, where any variable x in t has the property that it is linear in t and does not occur in any of the other subgoals. Thus, we can apply a proof quite similar to that in Theorem 3.6, and show that the procedure is terminating for this case.

The ordering that we use in this case compares the right hand sides of goals using a suitable property as before. In general it may not be possible to compare non-ground terms using such a criterion, but the reason that this still works in this case is because a non-decreasing function can not have variable dropping rules, and thus we end up comparing two non-ground terms in which the variables in one term (which has to be smaller with respect to this property) is a subset of the variables of the other one. □

There is another simple class of systems with variable dropping rules that have decidable semantic matching problems:

Theorem 4.6. *Let R be a rewrite system and P a suitable property. If*

- *all right hand sides for rules in R are either variables, or have a constructor at the top-level,*
- *all right hand sides are such that no defined function is nested below any decreasing function,*
- *all variable dropping rules in R are such that each of the dropped variables appear immediately below the top level symbol on the left hand side, and*
- *the left hand side is linear in each dropped variable,*

then the semantic matching problem is decidable for R.

Proof. Let $f(\ldots, x, \ldots) \rightarrow r$ be a rule in the system, such that x is a dropped variable. If a goal of the form $f(\ldots, s_i, \ldots) \rightarrow^? t$ is used in an application of the transformation rule **Mutate**, then one of the new goals generated (corresponding to the dropped variable x) would look like $s_i \rightarrow^? x$ (assuming that the variable x was the ith subterm of the left hand side term of the rule). Now, since we rename variables before applying a rule, x can not occur in any other term in the remaining multiset of goals. Thus, like in Lemma 4.5, we can ignore this goal, since it will be trivially solvable for any substitution for the variables of s_i.

This argument shows that any goal which has a dropped variable on its right hand side can be ignored. Thus, for the remaining goals we can apply arguments similar to those in Theorem 3.6, and hence the result. □

The following rewrite systems satisfy all the requirements of Theorem 4.3 and therefore have decidable semantic matching problems. Once again we have used depth as the suitable property.

Example 4.7. *In this example $+$ is depth non-decreasing, while $*$ is potentially depth decreasing, and has variable dropping rules.*

$$\begin{array}{rcl} 0 + x & \rightarrow & x \\ s(x) + y & \rightarrow & s(x + y) \end{array}$$

$$\begin{array}{rcl} 0 * x & \rightarrow & 0 \\ s(x) * 0 & \rightarrow & 0 \\ s(x) * s(y) & \rightarrow & s(y + (x * s(y))) \end{array}$$

Example 4.8. *In this example insert is depth increasing and both rules for it have constructors at the top-level on the right hand side, while min has variable dropping rules.*

$$\begin{array}{rcl}
min(x,0) & \to & 0 \\
min(0,x) & \to & 0 \\
min(s(x),s(y)) & \to & s(min(x,y)) \\
 & & \\
max(x,0) & \to & x \\
max(0,x) & \to & x \\
max(s(x),s(y)) & \to & s(max(x,y)) \\
 & & \\
sort(nil) & \to & nil \\
sort(cons(x,y)) & \to & insert(x,sort(y)) \\
 & & \\
insert(x,nil) & \to & cons(x,nil) \\
insert(x,cons(y,z)) & \to & cons(min(x,y),insert(max(x,y),z))
\end{array}$$

5 Related Results

Some results similar to those given here have been reported in [Hul80, KN87], where they are interested in the more general problem of semantic unification. Hullot [Hul80] shows that the *narrowing* procedure terminates when all right hand sides are either variables or ground terms. Furthermore, it has been demonstrated by Kapur and Narendran [KN87] that if each right hand side is either ground or a subterm of the left hand side, the unification problem for the corresponding theory is NP-complete.

Using techniques similar to those in this paper, and the *full* set of transformation rules in [Mit90], it is not hard to show that there is a strategy which is terminating for Hullot's case. The full system essentially gives additional decomposition rules necessary to handle cases when the right hand side of a goal $s \to^? x$ happens to be a variable. For the systems that we have considered so far, such a goal can either be ignored (in some left linear cases), or can be replaced by a different goal which doesn't have this property (for the variable preserving case). However, in the general case, such simplifications are not possible, and thus we require new transformation rules to handle such cases. Since both sides of a goal may contain variables the orderings we used in Section 3 do not decrease, having no a priori bound on the measure of the right side of goals.

It is possible to extend Hullot's systems somewhat:

Theorem 5.1. *If all right hand side of rules are either constructor terms or ground terms, then semantic unification is decidable.*

Proof. Consider r in the mutation rule. In the ground case, the goal $r \to^? t$ has a solution only if the normal form of r syntactically matches t. In the constructor case, r is free of defined functions, and $r \to^? t$ has a solution only if r and t can be syntactically unified. □

However, we can prove the following theorem from [KN87]:

Theorem 5.2. *Let R be a rewrite system in which every right hand side is a subterm of the corresponding left hand side. Then, the unification problem is decidable for R.*

Proof. The basic idea is that, since the full set of transformation rules [Mit90] simulates *innermost* rewriting in reverse (see [DS87, Mit90] for details), at any stage the right hand side

t of a goal of the form $s \rightarrow^? t$ must be irreducible for a solution to be feasible. Thus, if at any stage we have to mutate at a position which comes from the left hand side of a previous rule application, then the corresponding solution is reducible and can be ignored (the procedure will enumerate another solution which is equivalent but irreducible). Now, every right hand side of a rule in R is a subterm of the corresponding left hand side. Thus, based on the previous argument, we never need to further mutate the $r \rightarrow^? t$ subgoal generated from any application of **Mutate**, which implies that the solution tree generated from this subgoal is finite. □

6 Future Work

It would be interesting to develop the ideas of the previous section. Further restrictions on the system, such as having *completely defined* functions (a function f is said to be *completely defined* by a rewrite system R, if the normal form of any ground term containing f is a constructor-only term), may help because of the following result:

Lemma 6.1. *If R consists of only completely defined function symbols, then innermost narrowing is complete with respect to ground solutions.*

Proof. Observe that, in the final normal form, all the defined functions must be removed. Thus, we can apply rules at any position where a defined function occurs, and, in particular, to the innermost position. □

With this understanding, in certain cases it will be possible to have a decision procedure for semantic unification. The idea is to apply a minimal substitution to a subset of the variables in the goal, so that the terms are reducible and compare the top-level constructor generated along the two sides. If the subterms generated after decomposition happen to be "smaller" than the original goal terms in a well-founded ordering, then the system will be solvable.

It may also be possible to extend Theorem 4.3 somewhat, by allowing rules which are left non-linear, like in Theorem 4.6, after demonstrating that these non-linear variables are never eliminated by the procedure at a later stage, when using a variable dropping rule in **Mutate**. It may be possible to give sufficient conditions to check such properties by analyzing a graph of the terms of the rewrite system.

Acknowledgement

We thank Deepak Kapur and the referees for their helpful comments.

References

[Bo87] Alexander Bockmayr. A Note on a Canonical Theory with Undecidable Unification and Matching Problem. In *Journal of Automated Reasoning*, Vol 3, pages 379–381, 1987.

[DS87] Nachum Dershowitz and G. Sivakumar. Solving Goals in Equational Languages. In *Proceedings of the First International Workshop Conditional Term Rewriting System*, Orsay, France, July 1987. Vol. 308, pages 45–55, of *Lecture Notes in Computer Science*, Springer Verlag (1987).

[DJ90] Nachum Dershowitz and Jean-Pierre Jouannaud. Rewrite systems. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, chapter 6, pages 243–320, North-Holland, Amsterdam, 1990.

[Fay79] M. Fay. First-order unification in an equational theory. In *Proceedings of the Fourth Workshop on Automated Deduction*, pages 161–167, Austin, TX, February 1979.

[Hul80] Jean-Marie Hullot. Canonical forms and unification. In R. Kowalski, editor, *Proceedings of the Fifth International Conference on Automated Deduction*, pages 318–334, Les Arcs, France, July 1980. Vol. 87 of *Lecture Notes in Computer Science*, Springer, Berlin.

[HH87] Stephan Heilbrunner and Steffen Holldobler. The Undecidability of the Unification and Matching Problem for Canonical Theories. In *Acta Informatica*, Vol 24, pages 157–171, 1987.

[JK91] Jean-Pierre Jouannaud and Claude Kirchner. Solving Equations in Abstract Algebras: A Rule-Based Survey of Unification. In J.-L. Lassez and G. Plotkin, editors, *Computational Logic: Essays in Honor of Alan Robinson*, MIT Press, Cambridge, MA, 1991.

[KN87] Deepak Kapur and Paliath Narendran. Matching, Unification and Complexity. In *ACM SIGSAM Bulletin*, (1987) Vol. 21, Number 4, pages 6–9.

[Mit90] Subrata Mitra. Top-Down Equation Solving and Extensions to Associative and Commutative Theories. Master's thesis, Department of Computer and Information Sciences, University of Delaware, Newark, DE, 1990.

[NRS89] Werner Nutt, Pierre Réty and Gert Smolka. Basic Narrowing Revisited. In *J. of Symbolic Computation*, (1989) Vol. 7, pages 295–317.

[Ret87] Pierre Réty. Improving basic narrowing techniques. In P. Lescanne, editor, *Proceedings of the Second International Conference on Rewriting Techniques and Applications*, pages 228–241, Bordeaux, France, May 1987. Vol. 256 of *Lecture Notes in Computer Science*, Springer, Berlin.