Semantic Matching in Rewrite Theories

Subrata Mitra
IBM Software Solutions Division

and

Nachum Dershowitz
Hebrew University

P.O. Box 14582
Jerusalem 91145 Israel
phone/fax [+972] (2) 627-3440
nachum@cs.uiuc.edu

January 1997

Abstract

"Semantic matching" is the process of generating a basis set of substitutions (of terms for variables) that makes one term equal to another in a specified theory. We restrict ourselves here to matching problems in equational theories that can be presented as programs in the form of convergent rewrite systems, that is, finite sets of equations that compute unique output values when applied (from left-to-right) to input values (a generalization of functional programs).

Decidable matching can help in program verification and synthesis. We describe a new class of programs for which matching is decidable, which—with some negative results—provide a finer characterization of decidability than was available before.

1 Introduction

Equation solving is the process of finding a substitution (of terms for variables) that makes two terms equal in a given theory, while *semantic unification* is the process which generates a basis set of such unifying substitutions. For any solution to a given goal, the basis set must contain an element that is equivalent in the underlying theory to one that is at least as general. A simpler version of this problem, *semantic matching*, restricts the substitution to apply only to one of the terms (called the *pattern*). While semantic unification is used in some theorem provers for performing deductions modulo an equational theory (the theory of associativity and commutativity is a prime example), semantic matching has potential applications in pattern-directed languages and in their verification. For example, in a functional language, we may define *append* (@) and *reverse* (r) on lists (constructed with : and ϵ) using the following equations:

$$\begin{array}{rclcrcl} \epsilon @ x & = & x & & r(\epsilon) & = & \epsilon \\ (x:y)@z & = & x:(y@z) & & r(x:y) & = & r(y)@(x:nil) \end{array}$$

Given these equations, we might wish to ask questions such as whether the reverse of a non-empty list can be empty. Or, for the system

$$\begin{array}{rclcrcl} pop(x:y) & = & x & & pop(\epsilon) & = & \bot \\ push(x,\epsilon) & = & x:\epsilon & & pop(\bot) & = & \bot \\ push(x,y:z) & = & x:(y:z) & & push(x,\bot) & = & \bot \end{array}$$

one may ask whether $pop(push(x,y))$ can yield $\bot$ for any x and y. To answer such questions a matching algorithm is appropriate.

A rewrite system is *convergent* (technically, *ground* convergent) if every ground term has exactly one normal form. For such systems, every reducible substitution is equivalent in the theory to an irreducible one; hence, one can ignore reducible solutions to semantic unification and matching problems. For example, for a goal like $(1:x)@(2:\epsilon) \stackrel{?}{=} 1:2:\epsilon$, in the theory of append, the only solution of interest is $x \mapsto \epsilon$ (and not $x \mapsto \epsilon@\epsilon$, and so forth). It is well-known that any strategy for finding a complete set of matchings 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; see, for example, [Heilbrunner and Hölldobler, 1987; Bockmayr, 1987]. On the other hand, for some special classes of theories—associativity, for instance—semantic matching is decidable.

In this paper, we are interested in matching in theories that have a convergent presentation. Since the general matching problem with convergent systems is known to be undecidable, we are interested in characterizing restricted convergent systems (using syntactic criteria) for which either the matching problem is decidable, which constitutes the positive cases (i.e., restrictions over and above convergence results in decidability), or or for which it remains undecidable, which constitutes negative results. As such, we are not interested in specific theories such as associativity and commutativity. Rather, our aim is to be able to characterize classes of rewrite systems with a decidable matching problem.

Given a set $\mathcal{F}$ of function symbols and a (denumerable) set $\mathcal{X}$ of variables, the set of (first-order) *terms* $\mathcal{T}(\mathcal{F},\mathcal{X})$ is the smallest set containing $\mathcal{X}$ such that $f(t_1,\ldots,t_n)$ is in $\mathcal{T}(\mathcal{F},\mathcal{X})$ whenever $f \in \mathcal{F}$ and $t_i \in \mathcal{T}(\mathcal{F},\mathcal{X})$ for $i = 1,\ldots,n$. A term t is said to be *linear* in a variable x if x occurs exactly once in t, while a term is *linear* if it is linear with respect to each of its variables, for example, $x + (s(y) \times z)$. The *depth* of a term is the number of nodes in the longest path in its tree representation (so a constant or variable has depth one). A *substitution* σ is a special kind of replacement operation, uniquely defined by a mapping from variables to terms which is equal to identity almost everywhere, and written out as $\{x_1 \mapsto s_1,\ldots,x_m \mapsto s_m\}$. If t is a term containing variables, the $t\sigma$ is t with each occurrence of x_i replaced by s_i.

A *rewrite rule* is an ordered equation between terms, written as $l \to r$, for terms l and r. A rule is (*left-*) *right-linear* is its (left-) right-hand side is linear, it is *linear* if it is both left- and right-linear, and is *non-erasing* if every variable in l also appears in r. A *rewrite system* is a finite set of rewrite rules. We use $\to$ to denote a single step of derivation (application of a rewrite rule), and $\to^*$ as its reflexive-transitive

closure. A term s is said to be *irreducible* or in *normal form* if there is no term t such that $s \to t$. We write $s \to^! t$ if $s \to^* t$ and t is in normal form, and we say that t is the normal form of s. A rewrite relation $(\to)$ is *terminating* if there exists no infinite chain of rewrites of the form $t_1 \to t_2 \to \ldots \to t_k \ldots$. A rewrite relation is (*ground*) *confluent* if, whenever $u \to^* s$ and $u \to^* t$ there is a term v such that $s, t \to^* v$. A rewrite system which is both terminating and (ground) confluent is said to be (*ground*) *convergent*. Convergence is a reasonable requirement for most functional programs. In this paper we only concern ourselves with such systems.

It is sometimes convenient to partition $\mathcal{F}$ into two disjoint sets: *defined functions* and *constructors*. For our purpose any function symbol that appears at the top of the left-hand side of a rule is defined, while all others are constructors. A term is said to be *flat* if it has at most one defined function with no function symbol nested below the defined function. Also, we will say that a rewrite system has a property (for example, left-linearity) if each of its rules has the said property.

A matching goal is written as $s \to^? N$ (where N is a normal form) and has a solution σ if $s\sigma \to^! N$. For convergent systems, in general, semantic matching is as difficult as unification. For example, solving the unification goal $s \stackrel{?}{=} t$ in a convergent theory R is equivalent to solving the goal $eq(s,t) \to^? true$ in the theory $R \cup eq(x,x) \to true$. For *non-erasing* and *left-linear* rewrite systems matching is simpler than unification; we show how to match in Section 3. However, these restrictions themselves do not suffice for decidable matching in such theories, and in Section 4 we introduce additional restrictions on the rules such that matching becomes decidable, and show that each restriction is necessary. We start in Section 2 by showing that in general matching is undecidable, even for linear flat systems.

Missing definitions can be found in the survey [Dershowitz and Jouannaud, 1990].

2 Undecidable Matching

Linearity and flatness are simple, easy-to-check syntactic restrictions on rewrite rules. Such restrictions have been used extensively in the study of properties such as modular termination and confluence of rewrite systems. It is known that matching is undecidable even in convergent theories presented as (left- and right-) linear rewrite systems [Heilbrunner and Hölldobler, 1987]. On the other hand, matching is decidable if the system is left-flat [Christian, 1992]. However, rewrite systems with flat left-hand sides are truly restrictive, and do not allow any recursively defined functions.

We show below that matching is undecidable in the right-flat linear case. Although linearity and flat right-hand sides do not suffice, matching becomes decidable with the additional restriction that for each defined function there is at most one rule with a non-constructor right-hand side, and that side must be flat and have only one occurrence of defined functions [Dershowitz and Mitra, 1993].

Theorem 1. *There is no decision procedure for the matching problem in a (left- and right-) linear convergent rewrite system, in which every right-hand side is flat.*

Proof. We reduce semantic matching in such theories to the undecidable Post Correspondence Problem (PCP).

An instance of PCP consists of two lists $A = w_1, \ldots, w_k$ and $B = x_1, \ldots, x_k$, of strings over some alphabet Σ. This instance has a *solution* if there exists a sequence of integers $i_1, \ldots, i_m, m \geq 1$, such that

$$w_{i_1}, \ldots, w_{i_m} = x_{i_1}, \ldots, x_{i_m}.$$

The following example illustrates how semantic matching can be used to generate solutions to a particular instance of the Post Correspondence Problem:

Example 1. Let $\Sigma = \{s, p\}$, while $A = (w_i)_i$ and $B = (x_i)_i$ are as given below:

i	w_i	x_i
1	s	sss
2	$spsss$	sp
3	sp	p

We construct the following convergent rewrite system R:

$$\begin{array}{rcl} eq(s(x), s(y)) & \to & eq(x, y) \\ eq(p(x), p(y)) & \to & eq(x, y) \\ \\ A(\epsilon) & \to & \epsilon \\ A(1:x) & \to & s(A(x)) \\ A(2:x) & \to & s(p(s(s(s(A(x)))))) \\ A(3:x) & \to & s(p(A(x))) \\ \\ B(\epsilon) & \to & \epsilon \\ B(1:y) & \to & s(s(s(B(y)))) \\ B(2:y) & \to & s(p(B(y))) \\ B(3:y) & \to & p(B(y)) \end{array}$$

It is easy to see that this instance of PCP has a solution if and only if the matching goal

$$eq(A(x:y), B(x:y)) \stackrel{?}{\to} eq(\epsilon, \epsilon)$$

is satisfiable.

From the construction, it is evident that, given any instance of PCP, we can similarly construct a convergent rewrite system with the required syntactic restrictions, such that the matching problem described above has a solution if and only if the instance of PCP under consideration has one. The proof of convergence of the resulting rewrite system is based on the fact that no two left-hand sides unify, and that the system is terminating which can be shown using the recursive path ordering with the following precedence: $A > B > s > p > \epsilon$. Therefore, a decision procedure for matching (and thus unification) in such rewrite systems could be used to decide the Post Correspondence Problem. □

3 Complete Matching

For convergent systems, the unification (and therefore matching) problem is recursively enumerable. In other words, there exists a procedure that can find a unifier (match) whenever one exists. See [Jouannaud and Kirchner, 1991] for a survey of unification.

If we restrict ourselves to convergent rewrite systems that are, additionally, either non-erasing or left-linear, then the non-deterministic transformation rules of Table 1 constitute a complete set for the matching problem:

Theorem 2 (Completeness). *Let $\mathcal{R}$ be either a left-linear or a non-erasing convergent rewrite system. If the goal $s \to^? N$ has a solution θ (that is, $s\theta \to^! N$, for normal form N), then there is a derivation of the form $\{s \to^? N\} \stackrel{!}{\rightsquigarrow} \mu$, such that μ is a substitution at least as general as θ.*

Proof. In full version. □

For non-erasing systems, Bind can be further simplified, as shown in [Mitra, 1994].

Eliminate	$\{x \to^? t\}$
	$\rightsquigarrow$
	$\{x \mapsto t\}$
	where x is a free-variable that does not occur in t
Bind	$\{x \to^? s, x \mapsto t\}$
	$\rightsquigarrow$
	$x \mapsto s, mgu(s,t)$
	if x does not occur in s
Mutate	$\{f(s_1,\ldots,s_n) \to^? t\}$
	$\rightsquigarrow$
	$\{s_1 \to^? l_1, \ldots, s_n \to^? l_n, r \to^? t\}$
	where $f(l_1,\ldots,l_n) \to r$ is a renamed rule in R
Decompose	$\{f(s_1,\ldots,s_n) \to^? f(t_1,\ldots,t_n)\}$
	$\rightsquigarrow$
	$\{s_1 \to^? t_1, \ldots, s_n \to^? t_n\}$

Table 1: Transformation rules for semantic matching with left-linear or non-erasing convergent systems

4 Decidable Matching

For convergent systems, in general, semantic matching is as difficult as semantic unification. For example, solving the goal $s \stackrel{?}{=} t$ in a convergent theory R is equivalent to solving the goal $eq(s,t) \to^? true$ in the theory $R \cup eq(x,x) \to true$, for a new function symbol eq and constant $true$; the augmented theory is convergent since eq is a new symbol, not in R. A natural question is: Under what conditions is matching decidable (independent of unification)? The rewrite system constructed to simulate PCP in Example 1 is linear and non-erasing; therefore, linearity and non-erasing are not enough to guarantee decidability of matching.

Theorem 3. *Let $\mathcal{R}$ be a convergent left-linear rewrite system. If for every rule $f(l_1,\ldots,l_n) \to r$ in $\mathcal{R}$*

1. *each $l_i, 1 \le i \le n$, is of depth at most two,*
2. *r is either a variable or has a constructor at the root, and*
3. *whenever at least one l_j has depth greater than one, r has depth greater than one,*

then the semantic matching problem is decidable for $\mathcal{R}$.

Proof. In appendix. □

Example 2. The following definition of squaring using $+$ and $\times$ obeys all the syntactic restrictions of Theorem 3, and therefore has a decidable matching problem:

$$
\begin{aligned}
0 + x &\to x && (1)\\
s(x) + y &\to s(x + y) && (2)\\
0 \times x &\to 0 && (3)\\
x \times 0 &\to 0 && (4)\\
s(x) \times s(y) &\to s(y + (x \times s(y))) && (5)\\
sq(0) &\to 0\\
sq(s(x)) &\to s(sq(x) + (s(s(0)) \times x))
\end{aligned}
$$

Example 3. As another example, consider inserting a number in its correct place, in a list of numbers:

$$\begin{aligned} min(x,0) &\rightarrow 0 \\ min(0,x) &\rightarrow 0 \\ min(s(x),s(y)) &\rightarrow s(min(x,y)) \end{aligned}$$

$$\begin{aligned} max(x,0) &\rightarrow x \\ max(0,x) &\rightarrow x \\ max(s(x),s(y)) &\rightarrow s(max(x,y)) \end{aligned}$$

$$\begin{aligned} insert(x,\epsilon) &\rightarrow x:\epsilon \\ insert(x,y:z) &\rightarrow min(x,y):insert(max(x,y),z) \end{aligned}$$

Each of the restrictions in Theorem 3 is necessary for decidability: If we drop the requirement of left-linearity, then we get undecidability by encoding unification for left-linear systems as a matching problem for non-left-linear systems. In the remaining cases, we show that matching of certain goals would result in unification in the theories of addition (+) and multiplication ($\times$). (Notice that the definitions of + and $\times$ in Example 2 obey all the syntactic restrictions of Theorem 3. Thus, the matching problem is decidable for this system. However, its unification problem is undecidable, due to the undecidability of the Hilbert's Tenth Problem.)

Example 4. First we relax Condition 3, that is, we allow subterms of depth greater than one below the root on the left-hand side, without requiring that the right-hand side be of depth at least two. Consider the rewrite system consisting of the rules for addition and multiplication (rules 1–5) together with the following rewrite rules (the collective system can be proved convergent):

$$f(1) \rightarrow 1 \tag{6}$$
$$f(s(1)) \rightarrow 1 \tag{7}$$

$$g(1,1) \rightarrow s(1) \tag{8}$$
$$g(s(x),s(y)) \rightarrow s(f(g(x,y))) \tag{9}$$

Rule 7 is the only one which violates Condition 3.

We have $g(x,y) = s(1)$ if and only if $x = y = s^n(1)$ (in the theory of + and $\times$) for some $n \geq 0$. We have

$$g(s^{n+m}(1),s^n(1)) \rightarrow^n (sf)^n(g(s^m(1),1)) \rightarrow^{m=0} (sf)^n(s(1)) \rightarrow^n s(1)$$

Theorem 4. *The matching problem is undecidable if Condition 3 does not hold.*

Proof. Suppose t and t' are general terms involving + and $\times$ alone. Therefore, a goal of the form $g(t',t) \rightarrow^? s(1)$, would, in general, be undecidable, since a decision procedure for this problem could be used to solve Hilbert's Tenth Problem, which is impossible. □

Example 5. To relax Condition 2, by allowing defined functions to appear as the root of the right-hand sides, consider the convergent rewrite system consisting of the rules for addition and multiplication, together with the following additional rewrite rules:

$$f(1) \rightarrow 1 \tag{10}$$

$$g(1,1) \rightarrow 1 \tag{11}$$
$$g(s(x),s(y)) \rightarrow f(g(x,y)) \tag{12}$$

Rule 12 is the only one that violates Condition 2. We have:

$$g(x,y) = 1 \text{ if and only if } x = y = s^n(1), n \geq 0.$$

Theorem 5. *The matching problem is undecidable if Condition 2 does not hold.*

Example 6. Finally, we relax Condition 1, and allow depths greater than two below the root function on left-hand sides of rules (but in order to make sure that the last condition not be violated, we would insist that whatever depth we have on the left-hand side must show up on every path on the right-hand side, by way of leading constructors). We encode f from Example 4 using new rules:

$$\begin{aligned} F(s(x)) &\rightarrow s(1) && (13)\\ G(s(s(1)), s(s(x))) &\rightarrow s(s(s(x))) && (14)\\ \\ g(1,1) &\rightarrow s(s(1)) && (15)\\ g(s(x), s(y)) &\rightarrow s(F(G(g(x,y), s(s(1))))) && (16) \end{aligned}$$

None of the rules have an immediate subterm on the left-hand side that is of greater depth than the depth of the corresponding right-hand side. However, Rule 13 erases x; while Rule 14 is the only one that violates the depth criterion for left-hand sides (it allows immediate subterms of depth 3 on its left-hand side). We have $F(G(s(s(1)), s(s(x)))) \rightarrow F(s(s(s(x)))) \rightarrow^! s(1)$, and

$$g(x,y) = s(s(1)) \text{ if and only if } x = y = s^n(1), n \geq 0.$$

Theorem 6. *The matching problem is undecidable if Condition 1 does not hold.*

5 Conclusion

Semantic matching is useful for verifying some properties of functional programs, for incorporating logic-programming capabilities in a functional language, for constraint based systems and for theorem proving, in general. Even when a system admits a convergent presentation, the usual case in verification, the corresponding unification or matching procedure may be undecidable.

We have studied restricted convergent systems for which matching is decidable. This result complements one given in [Dershowitz *et al.*, 1992] for restricted non-erasing systems. The main difference between the two is that in this paper we use the purely syntactic property of depth, while [Dershowitz *et al.*, 1992] used a semantic property. Recently [Aguzzi and Modigliani, 1994] have extended the decidability criterion of [Dershowitz *et al.*, 1992] by introducing the notion of *positional-increase* to replace *increase*. By using positional information, it is possible to handle certain rewrite systems that does not have leading constructors on the right-hand sides of rules (for example, the usual presentation of $\times$ contains rules $\{x \times 0 \rightarrow 0, s(x) \times y \rightarrow y + (x \times y)\}$, instead of the 3 rules of Example 2). We believe that a similar refinement (positional depth) is possible for the positive result in this paper. With this extension we should be able to handle insertion sort by adding the following rules to those of Example 3: $sort(\epsilon) \rightarrow \epsilon$, $sort(x : y) \rightarrow insert(x, sort(y))$.

Linearity and flatness of rewrite rules are common syntactic properties which have been used for many different characterizations of rewrite systems (for example, confluence and termination). In this paper we have shown how these criteria affect matching. In most cases the results turn out to be negative, as was the case with the new result of this paper. However, with this new information, we have been able to complete the characterization of matching for systems with these two syntactic restrictions. Comon et al. [1991], use the idea of proving termination of a system of transformation rules for characterizing decidable unification, but with no assumption of convergence. In their case, both sides of the given equations must satisfy a slightly different non-nesting requirement than that of flatness as described here. (Since equations can be used in either direction, it is intuitive to use the restriction on both sides.) The resulting theories are simpler than the ones for convergent systems with flat right-hand sides (in fact, they bear similarity to the systems that are convergent and left-flat, for which a positive result was proved in [Christian, 1992]). Other characterizations of decidable unification appear in [Kapur and Narendran, 1987] (every right-hand side of a rule must be a proper subterm of the corresponding left-hand side) and [Hullot, 1980] (every right-hand

side must be a variable or a ground term). Unfortunately, none of these systems are powerful enough to capture truly recursive functions. Decidability results for unification in convergent systems was extended in [Dershowitz and Mitra, 1993], where the problems considered could potentially have infinite solutions, which were captured as indexed terms, along the lines of [Comon, 1992]. The important difference between the requirements of [Dershowitz and Mitra, 1993] and the one presented here is that we allow multiple flat terms with defined functions as right-hand sides for rules defining a given function. For instance, the definition of eq, in Example 1, used two rules, each of which has eq (a defined function) on its right-hand side, which would not be allowed by [Dershowitz and Mitra, 1993].

Appendix

The proof of completeness of the rules for matching (in the full version; see [Mitra, 1994]) uses a particular selection strategy for picking which subgoals to solve: after mutation, we always solve the $r \to^? t$ subgoal first, before solving the $s_i \to^? l_i$ in any order; after decomposition we could solve the new subgoals in any order. We presuppose that selection strategy here.

Lemma 7. *The most general unifier computed in Bind need only deal with linear terms.*

Proof. We may have to use bind because the rewrite rules may have non-linear variables on its right-hand sides, and since the left-hand side of the starting goal may be non-linear.

Suppose we start with the goal $S \to^? N$, for a ground normal-form N, use the selection strategy mentioned before, and apply transformation rules. Furthermore, consider the sequence of transformation rules before the first application of Bind. In this sequence, the generated substitutions should have been produced using Eliminate alone. Therefore, each term bound to a variable in this substitution must be linear (such terms must have come from N or from the left-hand sides of applied rules). Therefore, both s and t of Bind must be linear, and of independent variables. Thus, the computed *mgu* would again be a linear term. Furthermore, any of the linear variables in either s or t that gets bound during the mgu computation can be removed, since they do not appear anywhere else in either goals or substitutions, due to left-linear rules and the selection strategy. □

We now state a proof of Theorem 3:

Proof. Due to left-linearity, we only need to solve goals of the form $s \to^? t$, where any variable x in t is linear in t and does not occur in the right-hand side of any other subgoal (this is true because we have directed goals, and terms on the right-hand sides of goals could either be left-hand sides of (left-linear) rules from previous mutations, or subterms of the ground term N, when solving for a initial goal of the form $s' \to^? N$).

Let $\succ$ be the well-founded ordering on goals such that $s_1 \to^? t_1 \succ s_2 \to^? t_2$ if either:

- $depth(t_1) > depth(t_2)$, or
- $depth(t_1) = depth(t_2)$ and s_2 is a proper subterm of s_1.

The proof proceeds by picking a subgoal (say $s \to^? t$) from the remaining ones (following the selection strategy mentioned before). We show by induction on the multiset extension of the ordering $\succ$ that any solution to this goal is bounded in depth by that of t, and every application of a transformation rule decreases the complexity of goals in this ordering.

If the goal is of the form $x \to^? t$ (x being a variable) then there are two cases, either Eliminate applies (in which case the proposition is trivially true, since the goal gets removed from the collection and a binding gets added, but our ordering does not consider bindings) or Bind applies. In the latter case, given Lemma 7, we need only compute the most general unifier of two linear terms with independent variables. Therefore, the computed mgu is linear; furthermore, the depth is bounded by that of the deeper of the two terms for which the mgu is being computed. For the other cases, let $s = f(s_1, \ldots, s_n)$ and $t = N[\bar{x}]$. (We use the

notation $N[\bar{x}]$ to denote a term linear in variables $\bar{x}$ and for which no other subgoal in the current set has any of these variables on the right-hand side; it can be shown that considering such goals is sufficient.) There are several cases to be considered:

- If $N[\bar{x}]$ is a variable, then it can be shown that we do not have to solve this goal any further. Therefore, the only solution to this goal is an indeterminate (unbound variable) for each variable of $f(s_1, \ldots, s_n)$. Thus, the solution is of depth one, which is the same as that of $N[\bar{x}]$.

- If $N[\bar{x}]$ is a constant, then for decomposition to work, $f(s_1, \ldots, s_n)$ must be the identical constant, which gives the empty substitution as the only solution, and therefore the hypothesis holds in this case. Decrease in complexity is caused by the removal of the subgoal under consideration.

  Next, consider mutation of the goal $f(s_1, \ldots, s_n) \rightarrow^? N[\bar{x}]$, $N[\bar{x}]$ a constant, using a rule of the form $f(l_1, \ldots, l_n) \rightarrow r$. The only time such a rule could work is if $depth(r) = 1$. (For any other rule, by the assumption of the theorem, there has to be a constructor at the root of r, which would lead to failure when solving the $r \rightarrow^? N[\bar{x}]$ subgoal.) Furthermore, since r has depth one, by the assumption of the theorem, each $l_i, 1 \leq i \leq n$, must be of depth one also (Condition 3). If r is a constant (it also has to be a constructor, by Condition 2), then the only possible solution to the goal $r \rightarrow^? N[\bar{x}]$ is the empty substitution ($\sigma = \{\}$). However, if r is a variable, say z, then this goal has a unique solution of depth one (the solution is $\sigma = \{z \mapsto N[\bar{x}]\}$). Therefore, the derivation looks like:

  $$f(s_1, \ldots, s_n) \rightarrow^? N[\bar{x}] \quad \rightsquigarrow^* \quad s_1 \rightarrow^? l_1\sigma, \cdots, s_n \rightarrow^? l_n\sigma, \sigma$$

  In either case, each of the subgoals $s_1 \rightarrow^? l_1\sigma, \cdots, s_n \rightarrow^? l_n\sigma$ is smaller than the original goal $f(s_1, \ldots, s_n) \rightarrow^? N[\bar{x}]$ (since each l_i is either a variable or a constant, thus $depth(l_i\sigma) = 1$), and the proposition follows by induction on these smaller subgoals.

- For any other case, the depth of $N[\bar{x}]$ is at least two; let $N[\bar{x}] \equiv g(N_1, \ldots, N_m)$. Were we to decompose the goal, then each of the subgoals thus generated would be smaller in the ordering $\succ$. Thus, for decomposition, the proposition holds by induction on each of the smaller subgoals. Finally, consider mutation of this goal using a rule of the form $f(l_1, \ldots, l_n) \rightarrow r$:

  $$f(s_1, \ldots, s_n) \stackrel{?}{\rightarrow} N[\bar{x}] \rightsquigarrow_{\mathbf{Mutate}} s_1 \stackrel{?}{\rightarrow} l_1, \cdots, s_n \stackrel{?}{\rightarrow} l_n, r \stackrel{?}{\rightarrow} N[\bar{x}],$$

  there are further cases:

  - If we require r to be of depth one, then r must be a variable, say z. (A constant for r does not work, since the constant must be a constructor by the requirements of the theorem, and therefore, the goal $r \rightarrow^? N[\bar{x}]$ has no solution.) In this case the subgoal $r \rightarrow^? N[\bar{x}]$ (that is, $z \rightarrow^? N[\bar{x}]$) is trivially solvable, and the solution is bounded in depth by that of $N[\bar{x}]$. Furthermore, by the assumption of the theorem, each $l_i, 1 \leq i \leq n$, has depth one (given Condition 3, since we assumed r to be of depth one). Suppose σ is the (unique) solution to the goal $z \rightarrow^? N[\bar{x}]$. Therefore, as in the previous case, we have $depth(l_i\sigma) \leq depth(N[\bar{x}]), 1 \leq i \leq n$. Thus, the proposition holds by applying the hypothesis on the smaller subgoals $s_1 \rightarrow^? l_1\sigma, \cdots, s_n \rightarrow^? l_n\sigma$.
  - If r has depth greater than one, then it must have a leading constructor (by assumption of the theorem). Suppose $r \equiv g(r_1, \ldots, r_m)$, where g is a constructor (if the root function of r is different from g, then we get failure, so this is the only case to be considered). Thus, it is possible to decompose the goal $r \rightarrow^? N[\bar{x}]$ at least once, leading to smaller subgoals of the form $r_1 \rightarrow^? N_1, \ldots, r_m \rightarrow^? N_m$ (that is, $f(s_1, \ldots, s_n) \rightarrow^? N[\bar{x}] \succ r_i \rightarrow^? N_i, 1 \leq i \leq m$). Furthermore, let $depth(N[\bar{x}]) = d \geq 2$, which means that $max(depth(N_i)) = d - 1, 1 \leq i \leq m$. Although we could solve these subgoal in any order, for clarity of presentation, let us assume that we solve them in left-to-right sequence. In other words, we first solve $r_1 \rightarrow^? N_1$, to get a solution σ_1 (which, by inductive hypothesis, must be bounded in size by $d - 1$). Next, we solve $r_2 \rightarrow^? N_2\sigma_1$. Due to

the linearity requirements on $N[\bar{x}]$, $N_2\sigma_1 \equiv N_2$, and therefore, we could still use the inductive hypothesis on this goal. Eventually, we would solve $r_m \rightarrow^? N_m\sigma_{m-1}$, where σ_{m-1} is the collective solution from solving $r_i \rightarrow^? N_i, 1 \leq i \leq m-1$, and we again apply the same technique, to get the solution σ_m, which again should be bounded by $d-1$. Notice that σ_m is indeed a solution to $r \rightarrow^? N[\bar{x}]$: therefore, we have demonstrated that any solution to $r \rightarrow^? N[\bar{x}]$ would be bounded in depth by $d-1$. Let σ be such a solution, which gives us the new set of subgoals as:

$$f(s_1, \ldots, s_n) \rightarrow^? N[\bar{x}] \quad \rightsquigarrow^* \quad s_1 \rightarrow^? l_1\sigma, \cdots, s_n \rightarrow^? l_n\sigma, \sigma$$

where $l_i\sigma, 1 \leq i \leq n$, is bounded in depth by d (only those terms which contain variables become deeper after instantiation; however, when we substitute a variable, which could occur at at most below one function, by a term bounded in depth by $d-1$, we could get a term of depth at most d). Thus, each of the new subgoals is smaller (in $\succ$) than the original, and the proposition follows by induction on these smaller goals.

□

References

[Aguzzi and Modigliani, 1994] G. Aguzzi and U. Modigliani. A Criterion to Decide the Semantic Matching Problem. In *Proceedings of the International Conference on Logic and Algebra (in memory of Prof. Magari)*, Italy, 1995.

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

[Christian, 1992] J. Christian. Some Termination Criteria for Narrowing and E-Narrowing. In *Proceeding of the Eleventh International Conference on Automated Deduction*, Saratoga Springs, New York, June 1992. Volume 607, pages 582–588, of *Lecture Notes in Artificial Intelligence*, Springer-Verlag.

[Comon, 1992] H. Comon. On unification of terms with integer exponents. Technical Report 770, Universite de Paris-Sud, Laboratoire de Recherche en Informatique, 1992.

[Comon *et al.*, 1991] H. Comon, M. Haberstrau, and J.-P. Jouannaud. Decidable problems in shallow equational theories. Technical Report 718, Universite de Paris-Sud, Laboratoire de Recherche en Informatique, December 1991.

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

[Dershowitz *et al.*, 1992] N. Dershowitz, S. Mitra, and G. Sivakumar. Decidable matching for convergent systems. In *Proceedings of the Eleventh Conference on Automated Deduction*, pages 589–602, Saratoga Springs, NY, June 1992. Vol. 607 of *Lecture Notes in Artificial Intelligence*, Springer-Verlag, Berlin.

[Dershowitz and Mitra, 1993] N. Dershowitz and S. Mitra. Higher-order and semantic unification. In *Proceedings of the Thirteenth International Conference on Foundations of Software Technology and Theoretical Computer Science*, Bombay, India, 1993. Volume 761, pages 139–150, of *Lecture Notes in Computer Science*, Springer Verlag.

[Heilbrunner and Hölldobler, 1987] S. Heilbrunner and S. Hölldobler. The undecidability of the unification and matching problem for canonical theories. *Acta Informatica*, 24(2):157–171, April 1987.

[Hullot, 1980] J.-M. 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.

[Jouannaud and Kirchner, 1991] J.-P. Jouannaud and C. 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.

[Kapur and Narendran, 1987] D. Kapur and P. Narendran. Matching, unification and complexity (a preliminary note). *SIGSAM Bulletin*, 21(4):6–9, November 1987.

[Mitra, 1994] S. Mitra. Semantic Unification for Convergent Systems. PhD thesis, Department of Computer Science, University of Illinois, Urbana, IL, 1994. Appears as Dept. of Computer Science Technical Report Number UIUCDCS-R-94-1855, October, 1994.