J. Symbolic Computation (1995) **11**, 1–000

The STO-problem is NP-hard

KRZYSZTOF R. APT[†,‡], PETER VAN EMDE BOAS[‡] AND ANGELO WELLING[‡]

[†] *CWI, P.O. Box 94079, 1090 GB Amsterdam, The Netherlands*
[‡] *Dept. of Math. and Comp. Sci., University of Amsterdam, The Netherlands*

(Received 9 May, 1994)

A finite set of term equations E is called subject to the occur-check (STO) if a sequence of actions of the Martelli-Montanari unification algorithm starts with E and ends with a failure due to occur-check. We prove here that the problem of deciding whether E is STO is NP-hard.

1. Introduction

For efficiency reasons in most Prolog implementations the so-called occur-check is omitted from the unification algorithm. This naturally calls for a definition of unification without the occur-check and for a characterization of the sets of term equations for which this omission might be of importance for unification purposes. The latter has been offered by Deransart, Ferrand and Téguia (1991), who introduced the notion of a set of equations being STO (*S*ubject *T*o *O*ccur-check). Informally, a set of equations is STO if some sequence of actions of the nondeterministic Martelli-Montanari unification algorithm leads to a situation in which the failure due to the occur-check arises.

As the known unification algorithms - see, for example, Robinson [7, 8], Venturini-Zilli [10], Martelli and Montanari [5], Paterson and Wegman [6] - are special cases of the Martelli-Montanari algorithm, this concept describes when unification without the occur-check might lead to problems. This is apparently as close as one can get to a characterization of the sets of equations for which unification might depend on the presence of the occur-check. Therefore, not surprisingly, the definition of an STO set of equations entered the proposal for standard Prolog (see Scowen [9]).

The result of this paper indicates an unexpected difference between the two relevant properties of sets of equations. As was shown by Paterson and Wegman [6] the property of being unifiable can be tested in linear time. We prove that the property of being STO is NP-hard. Recall that a problem is NP-hard, if its solvability in polynomial time implies that every problem in the class NP is solvable in polynomial time. This shows that, for all practical purposes, the definition of standard Prolog refers to a computationally intractable concept.

So the "STO test" apparently cannot be efficiently implemented. A possible remedy could be to identify a more limited property than that of being STO, which could be efficiently implemented and which would still be able to capture the original intention that for a set of equations satisfying the property omission of the occur-check might

0747–7171/90/000000 + 00 $03.00/0

be of importance for unification purposes. Such a property could exploit some limited information available at the implementation level, for example that the equations are ordered. That is, it could be defined on sequences, rather than sets of equations.

2. Preliminaries

Throughout the paper, the symbol $\equiv$ (resp. $\not\equiv$) is used to indicate syntactic equality (resp. inequality), the set of variables occurring in any syntactic object O is denoted by $Var(O)$ and the arity of a function symbol f is denoted by $Arity(f)$. A function symbol of arity 0 is called a constant.

From now on we fix a finite set of function symbols F and a finite set of variables V. The class of *terms over F and V* is defined recursively as follows:

a variable is a term,
if $t_1, ..., t_n$ are terms, $f \in F$, $Arity(f) = n$, then $f(t_1, ..., t_n)$ is a term.

A *substitution* is a finite mapping from variables to terms which assigns to each variable x in its domain a term t different from x. We write it as

$$\{x_1/t_1, \ldots, x_n/t_n\}$$

where

$x_1, \ldots, x_n$ are different variables,
$t_1, \ldots, t_n$ are terms,
$x_i \not\equiv t_i$ for $1 \leq i \leq n$.

The application of a substitution to a (set of) term(s) and the relation *"more general than"* between the substitutions is defined in the usual way. A *set of equations* E is a finite set of the form $\{s_1 = t_1, s_2 = t_2, ..., s_n = t_n\}$, where s_i and t_i are terms, for $1 \leq i \leq n$. A substitution σ such that $s_1\sigma \equiv t_1\sigma, ..., s_n\sigma \equiv t_n\sigma$ is called a *unifier* of E. A unifier of E is called a *most general unifier* (in short: *mgu*) of E if it is more general than all unifiers of E. Finally, we denote by $|E|$ the number of equations in E.

The problem of deciding whether a set of equations has a unifier is called the *unification problem*. This problem was introduced and solved by Robinson [7] by providing a unification algorithm. For our purposes we need the following nondeterministic unification algorithm due to Martelli and Montanari [5] (and informally introduced by Herbrand [4]).

MARTELLI-MONTANARI ALGORITHM

Given a set of equations, choose any equation of a form indicated below and perform the associated action. If no action applies to any equation, stop with success.

(1) $E \cup \{f(s_1, ..., s_n) = g(t_1, ..., t_m)\} \rightarrow$ *fail: clash*
where $f \not\equiv g$
(2) $E \cup \{f(s_1, ..., s_n) = f(t_1, ..., t_n)\} \rightarrow E \cup \{s_1 = t_1, ..., s_n = t_n\}$
(3) $E \cup \{x = x\} \rightarrow E$
where $x \in V$
(4) $E \cup \{t = x\} \rightarrow E \cup \{x = t\}$

where $x \in V, t \notin V$
(5) $E \cup \{x = t\}$ $\rightarrow$ *fail: positive occur-check*
where $x \in V, t \notin V, x \in Var(t)$
(6) $E \cup \{x = t\}$ $\rightarrow E\{x/t\} \cup \{x = t\}$
where $x \in V, x \in Var(E), x \notin Var(t)$

Note that action (1) includes the case of two different constants and action (2) includes the case $c = c$ for every constant c which leads to deletion of such an equation. The condition $x \notin Var(t)$ in action (6) is called the *occur-check test.* The following result is due to Martelli and Montanari [5].

THEOREM 2.1. *The Martelli-Montanari algorithm always terminates. If the original set of equations E has a unifier, then the algorithm terminates with success and produces an mgu of E written in an equational form, and otherwise it terminates with failure.* □.

Deransart, Ferrand and Téguia (1991) introduced the following notion.

DEFINITION 2.2. *A set of equations E is* subject to the occur-check *(STO) iff a sequence of actions of the Martelli-Montanari algorithm starts with E and ends with action (5). E is* not subject to the occur-check *(NSTO) iff it is not STO.* □.

Intuitively, E is NSTO iff unification and unification without the occur-check coincide for E. By Theorem 2.1 if an execution of the Martelli-Montanari algorithm terminates with success, the initial set of equations is NSTO. On the other hand, if an execution of the algorithm terminates with failure, the initial set of equations may be NSTO or STO. Consider for example the sets $\{a = f(a)\}$ and $\{x = f(x)\}$ with a a constant. Moreover, for some sets of equations different executions of the algorithm can terminate with failure for different reasons. Consider for example the set $\{a = f(a), x = f(x)\}$.

Scowen [9] lists the requirements for a formal definition of unification within standard Prolog. One of them (see top of page 934), when properly formalized, states that unification is undefined if the original set of equations is STO.

We show in this paper that the problem of deciding whether a set of equations is STO (in short: *the STO-problem*) is NP-hard.

3. The STO-problem is NP-hard

The following lemma allows us to reduce the STO test to simpler sets of equations.

LEMMA 3.1. (STO)

(i) If $f \not\equiv g$, then $E \cup \{f(s_1, ..., s_n) = g(t_1, ..., t_m)\}$ is STO iff E is STO.
(ii) If $x \in V, x \notin Var(E) \cup Var(t)$, then $E \cup \{x = t\}$ is STO iff E is STO.
(iii) $E \cup \{f(s_1, ..., s_n) = f(t_1, ..., t_n)\}$ is STO iff $E \cup \{s_1 = t_1, ..., s_n = t_n\}$ is STO.
(iv) If $x \in V$, then $E \cup \{t = x\}$ is STO iff $E \cup \{x = t\}$ is STO.

PROOF. Properties (i) and (ii) are obvious whereas (iii) and (iv) were proved in Deransart and Maluszynski (1993). □

Properties (iii) and (iv) state that actions (2) and (4) of the Martelli-Montanari algorithm preserve the STO property. The same obviously holds for action (3). However, action (6) can affect the STO property. Indeed, for a variable x and a constant a the set $\{x = a, x = f(x)\}$ is obviously STO, whereas $\{x = a, a = f(a)\}$ is not.

Finally, property (i) states that deletion of the equation to which action (1) applies does not affect the STO property. The corresponding property obviously fails for action (5) – just consider the above set $\{x = a, x = f(x)\}$.

DEFINITION 3.2. *Given a set of equations E, we denote by $Stand(E)$ the set of equations which is obtained from E by applying as many times as possible actions (2) and (4) of the Martelli-Montanari algorithm and by deleting the equations according to the STO Lemma 3.1.(i) and 3.1.(ii).* □

This brings us to the following conclusion.

THEOREM 3.3. *E is STO iff $Stand(E)$ is STO.*

PROOF. By the STO Lemma 3.1. □

To reduce the STO test to still more "elementary" sets of equations we need to apply action (6) of the Martelli-Montanari algorithm. However, as just observed, this action can affect the STO property, so we need to be careful. First we introduce the following notion.

DEFINITION 3.4. *Consider a set of equations E. A subset E' of E is* closed within *E if for some variable $x \notin Var(E - E')$ all equations of E' are of the form $x = s$, where $x \notin Var(s)$.* □

For example, the set $E = \{x = f(y), x = y, z = f(u), y = a\}$ has two subsets closed within E: $\{x = f(y), x = y\}$ and $\{z = f(u)\}$. Note that $\{y = a\}$ is not closed within E since $y \in Var(E - \{y = a\})$.

Observe that when E' is closed within E, then only action (6) can be applied to an equation from E'. This brings us to the following definition.

DEFINITION 3.5. *Consider a set of equations E and a subset E' closed within E. Given an equation $e \in E'$ we denote by $Reduce(E', e)$ the set of equations obtained from E' by applying action (6) to the equation e.* □

LEMMA 3.6. *Consider a set of equations E and a subset E' closed within E. Then E is STO iff for some $e \in E'$ the set $(E - E') \cup Reduce(E', e)$ is STO.*

PROOF. ($\Rightarrow$) Suppose E is STO. Consider a sequence of actions which leads to action (5). If this sequence does not select (an instance of) an equation from E' somewhere, the same sequence can be applied to $E - E'$, so a fortiori to $(E - E') \cup Reduce(E', e)$, for each $e \in E'$. So suppose now that this sequence selects (an instance of) an equation $x = s$ from E'. Consider the first such selection. By the form of E' the performed action is then action (6).

Let E_1 be the resulting set of equations. Thanks to the fact that $x \notin Var(E - E')$ the actions preceding this selection of $x = s$ do not introduce new occurrences of x in the considered sets of equations. Consequently, E_1 can also be obtained from E by a transposed sequence of actions in which the equation $x = s$ is selected first and then the original sequence of actions up to the selection of $x = s$ is performed. Consequently for some $e' \in E'$ the set E_1 can be obtained from $(E - E') \cup Reduce(E', e')$, so $(E - E') \cup Reduce(E', e')$ is STO.

($\Leftarrow$) By the fact that for all $e \in E'$ the set E reduces to $(E - E') \cup Reduce(E', e)$ by action (6). □

Intuitively, this lemma states that to determine whether E is STO it is sufficient to limit one's attention to the sequences of actions which start with action (6) applied to an equation in a subset of E' which is closed within E. We are now in position to prove the desired result.

THEOREM 3.7. *The STO-problem is NP-hard.*

PROOF. We provide a reduction from the known NP-Complete Satisfiability Problem (see e.g. Garey and Johnson [3]) to the STO-problem. Let $U = \{u_1, u_2, ..., u_n\}$ be a set of variables and $C = \{c_1, ..., c_m\}$ be a set of clauses making up an arbitrary instance of the Satisfiability Problem. A set of equations E is constructed such that E is STO if and only if C is satisfiable. E is a union of n disjoint subsets $E_1, E_2, ..., E_n$. Each E_i consists of four equations; two of them are associated with u_i and two with $\bar{u}_i$, the complement of u_i.

First, we define a set V of variable symbols and a set F of function symbols over which the terms occurring in E are built:

$$V = \{x_i \mid 1 \leq i \leq n\} \cup \{z_j \mid 1 \leq j \leq m\},\ F = \{f^i, g^i \mid 1 \leq i \leq n\} \cup \{h\}.$$

Now, let C_i denote the set of clauses of C which contain an occurrence of u_i and $\bar{C}_i$ denote the set of clauses of C which contain an occurrence of $\bar{u}_i$.

The arity of h is independent of the form of the particular instance of Satisfiability and is equal to one, whereas the arities of f^i and g^i do depend on this form and are respectively equal to the number of clauses in C_i and to the number of clauses in $\bar{C}_i$. In the following, "+1" denotes the "increment modulo m" over the set $\{1, \ldots, m\}$, so $m + 1 = 1$.

We are now ready to define the sets $E_i, 1 \leq i \leq n$. Two terms, $s_{i,1}$ and $s_{i,2}$, are constructed with the function symbol f^i as the outer constructor. Suppose the k^{th} clause of C_i is c_j. Then the k^{th} argument of $s_{i,1}$ is z_j and the k^{th} argument of $s_{i,2}$ is $h(z_{j+1})$. So, informally,

$$s_{i,1} \equiv f^i(..., z_j, ...) \text{ and } s_{i,2} \equiv f^i(..., h(z_{j+1}), ...),$$

with z_j and $h(z_{j+1})$ being the k^{th} arguments of, respectively $s_{i,1}$ and $s_{i,2}$. C_i contributes to E_i two equations

$$x_i = s_{i,1} \text{ and } x_i = s_{i,2}.$$

In the same way as above two terms $t_{i,1}$ and $t_{i,2}$ are constructed using the function symbol g^i. $\bar{C}_i$ contributes to E_i two equations

$$x_i = t_{i,1} \text{ and } x_i = t_{i,2}.$$

As an example of this construction, consider the following instance of the Satisfiability Problem: $U = \{u_1, u_2\}, C = \{c_1, c_2\}$, with $c_1 = \{u_1, \bar{u}_2\}$ and $c_2 = \{u_1, \bar{u}_1, u_2\}$. It yields the following set of equations:

$$\begin{array}{ll} & \{x_1 = f^1(z_1, z_2),\ x_1 = f^1(h(z_2), h(z_1)),\ x_1 = g^1(z_2),\ x_1 = g^1(h(z_1))\} \\ \cup & \{x_2 = f^2(z_2),\ x_2 = f^2(h(z_1)),\ x_2 = g^2(z_1),\ x_2 = g^2(h(z_2))\}. \end{array}$$

Given a truth assignment $t : U \rightarrow \{T, F\}$ we denote below its restriction to the variable u_i by $t[i]$. Each subset E_i is closed within E, so applying Lemma 3.6 n times we get

E is STO iff there are $e_1 \in E_1, ..., e_n \in E_n$, such that $\bigcup_{i=1}^n Reduce(E_i, e_i)$ is STO.

Fix such a sequence $e_1 \in E_1, ..., e_n \in E_n$ of equations. By Theorem 3.3

$$\bigcup_{i=1}^n Reduce(E_i, e_i) \text{ is STO iff } Stand(\bigcup_{i=1}^n Reduce(E_i, e_i)) \text{ is STO.}$$

Now for some truth assignment $t : U \rightarrow \{T, F\}$, (namely the one defined by $t(u_i) =$ **if** $e_i \in \{x_i = s_{i,1}, x_i = s_{i,2}\}$ **then** T **else** F **fi**, $1 \leq i \leq n$)

$$Stand(\bigcup_{i=1}^n Reduce(E_i, e_i)) = \bigcup_{i=1}^n \{z_j = h(z_{j+1}) \mid c_j \text{ is true under } t[i]\}.$$

Indeed, for e_i equal to $x_i = s_{i,1}$ we have

$$Reduce(E_i, e_i) = \{x_i = s_{i,1}, s_{i,1} = s_{i,2}, s_{i,1} = t_{i,1}, s_{i,1} = t_{i,2}\}$$

and $Stand(\{x_i = s_{i,1}, s_{i,1} = s_{i,2}, s_{i,1} = t_{i,1}, s_{i,1} = t_{i,2}\}) = \{z_j = h(z_{j+1}) \mid c_j \in C_i\} = \{z_j = h(z_{j+1}) \mid c_j \text{ is true under } t[i]\}$. And similarly for e_i equal to one of the other three equations of E_i.

But for every truth assignment $t : U \rightarrow \{T, F\}$

$$\bigcup_{i=1}^n \{z_j = h(z_{j+1}) \mid c_j \text{ is true under } t[i]\} = \{z_j = h(z_{j+1}) \mid c_j \text{ is true under } t\}.$$

Now, $\{z_j = h(z_{j+1}) \mid c_j \text{ is true under } t\}$ is STO iff $\{z_j = h(z_{j+1}) \mid c_j \text{ is true under } t\} = \{z_j = h(z_{j+1}) \mid 1 \leq j \leq m\}$ iff all clauses of C are true under t. Thus E is STO iff C is satisfiable.

It is clear that the construction of E from C can be accomplished in polynomial time, as for each variable $u_i \in U$ at most m clauses have to be checked for the occurrences of u_i and $\bar{u}_i$. $\square$

It would be interesting to know whether the STO-problem is NP-complete. We suspect it is but did not succeed in proving it. Note that a naive implementation of the Martelli-Montanari algorithm may lead to exponential growth of the set of equations.

References

Deransart P., Ferrand G., Téguia M. (1991). *NSTO programs (not subject to occur-check).* In *V. Saraswat and K. Ueda, editors, Proceedings of the International Logic Symposium,*

533-547. The MIT Press, 1991.
Deransart P., Maluszynski J. (1993). *A Grammatical View of Logic Programming.* The MIT Press.
Garey M., Johnson D. (1979). *Computers and Intractibility. A Guide to the Theory of NP-Completeness.* Freeman, New York.
Herbrand J. (1930). *Recherches sur la théorie de la démonstration.* Thèse de l'Université de Paris, 1930. English translation in: *Jacques Herbrand: Logical writings.* Ed. W. Goldfarb. Harvard, 1971, 148.
Martelli A., Montanari U. (1982). *An efficient unification algorithm.* In *ACM Transactions on Programming Languages and Systems*, Vol. 4, No. 2,258 - 282.
Paterson M., Wegman M. (1976). *Linear unification.* In *Proceedings of the Symposium on the theory of computing.* ACM Special Interest Group for automata and computability theory, 181 - 186.
Robinson J. A. (1965). *A machine oriented logic based on the resolution principle.* In *Journal of the Association for Computing Machinery*, Vol. 12, No. 1, 23-41.
Robinson J. A. (1971). *Computational logic: The unification computation.* In *Machine Intelligence*, Vol. 6,63-72.
Scowen R.S. (1991). *An overview of Prolog standardization – progress, problems and solutions.* In K. Furukawa, editor, *Proceedings of the Eighth International Conference on Logic Programming*, 922–936, Paris, France,The MIT Press.
Venturini-Zilli M. (1975). *Complexity of the unification algorithm for first-order expressions.* In *Calcolo.* Vol. 12, No. 4, 361-372.