On the Undecidability of Second-Order Unification*

Jordi Levy
Institut d'Investigació en Intel·ligència Artificial
Consejo Superior de Investigaciones Científicas
Campus UAB, 08193 Bellaterra, Barcelona, Spain
email: levy@iiia.csic.es

Margus Veanes
Max-Planck-Institut für Informatik
Im Stadtwald, 66123 Saarbrücken, Germany
email: veanes@mpi-sb.mpg.de

*This work was partially supported by the project MODELOGOS (TIC97-0579-C02-01) funded by the CICYT, and the ESPRIT Basic Research Action CCL. This paper is based on preliminary results in [Levy 1998, Veanes 1998, Levy & Veanes 1998].

Proposed running head: Second-Order Unification

Corresponding author: Margus Veanes,
Max-Planck-Institut für Informatik,
Im Stadtwald,
66123 Saarbrücken,
Germany.
email: veanes@mpi-sb.mpg.de
phone: +49-681-9325 218
fax: +49-681-9325 299

Abstract

There is a close relationship between word unification and second-order unification. This similarity has been exploited, for instance, in order to prove decidability of monadic second-order unification, and decidability of linear second-order unification when no second-order variable occurs more than twice. The attempt to prove the second result for (non-linear) second-order unification failed, and lead instead to a natural reduction from simultaneous rigid E-unification to this second-order unification. This reduction is the first main result of this paper, and it is the starting point for proving some novel results about the undecidability of second-order unification presented in the rest of the paper.

We prove that second-order unification is *undecidable* in the following three cases: 1) each second-order variable occurs at most twice and there are only two second-order variables; 2) there is only one second-order variable and it is unary; 3) the following conditions (i-iv) hold for some fixed integer n: (i) the arguments of all second-order variables are ground terms of size $< n$, (ii) the arity of all second-order variables is $< n$, (iii) the number of occurrences of second-order variables is ≤ 5, (iv) there is either a single second-order variable, or there are two second-order variables and no first-order variables.

List of symbols

character	explanation
␣	a “blank” symbol: `\texttt{\char32}`

All other symbols used in this document are standard symbols from LaTeX.

1 Introduction

Second-order unification [Pietrzykowski 1973] and restricted forms thereof have been studied in a wide range of areas, such as natural language processing [Niehren, Pinkal & Ruhrberg 1997] and term rewriting [Comon 1993], among others. Although Goldfarb [1981] showed second-order unification to be undecidable in general, Makanin's [1977] decidability result of word unification and Schulz's [1991] extension of it have been used as key tools in establishing the decidability of some important special cases of second-order unification [e.g. Farmer 1988]. In *linear* second-order unification, terms in unifiers must have exactly one occurrence of each bound variable, and in *context unification*, in addition to this condition, all second-order variables are unary. The relationship between word unification and context unification becomes clear when we codify a word unification problem, such as

$$H \cdot a \cdot b \cdot G \stackrel{?}{=} G \cdot b \cdot a \cdot H,$$

where '$\cdot$' denotes string concatenation, as the context unification problem

$$H(f_a(f_b(G(c)))) \stackrel{?}{=} G(f_b(f_a(H(c)))).$$

A possible unifier of the word equation maps H to the empty string and G to the string a, and the corresponding unifier of the context equation maps H to $\lambda x\,.\,x$ and G to $\lambda x\,.\,f_a(x)$.

We show that second-order unification, in its unrestricted form, is intimately related to a fundamental decision problem, called simultaneous rigid E-unification [Gallier, Raatz & Snyder 1987], that occurs in automated reasoning in (both classical and intuitionistic) first-order logic with equality [Degtyarev, Gurevich & Voronkov 1996]. Basically, this relationship relies on a close correspondence between second-order unification problems on a certain form, and ground rewriting. Consider two ground terms s and t, and a system of n ground rules $l_1 \to r_1, \ldots, l_n \to r_n$. Then

$$s \stackrel{*}{\to}_{\{l_1 \to r_1, \ldots, l_n \to r_n\}} t$$

if and only if this second-order equation is unifiable (let `cons` and `nil` be new symbols):

$$F(l_1, \ldots, l_n, \mathtt{cons}(t, \mathtt{nil})) \stackrel{?}{=} \mathtt{cons}(s, F(r_1, \ldots, r_n, \mathtt{nil})).$$

The precise reasons why this relationship holds are explained in detail in Section 3.

Despite many similarities, word unification is decidable [Makanin 1977], second-order unification is undecidable [Goldfarb 1981], simultaneous rigid E-unification is undecidable [Degtyarev & Voronkov 1995], and, with the exception of some special cases [Comon 1993, Schmidt-Schauß 1995, Levy 1996, Schmidt-Schauß 1998, Schmidt-Schauß & Schulz 1998], the question is open for linear second-order unification and context unification (although these problems are

conjectured to be decidable). Decidability of word unification was an open question for a long time and its proof [Makanin 1977] involves a lot of technicalities. However, it is very easy to prove that it is decidable when no variable occurs more than twice in a problem. The same main ideas were used to prove that linear second-order unification and context unification are decidable under the same restriction [Levy 1996]. However, we prove, in Section 4, that this is not the case for second-order unification, by reduction from simultaneous rigid E-unification.

We use properties underlying this reduction and additional ideas that have been used to prove the undecidability of restricted cases of simultaneous rigid E-unification [Plaisted 1995, Gurevich & Veanes 1999] and second-order unification [Farmer 1991], that second-order unification is undecidable in the following three cases: (Section 4) each second-order variable occurs at most twice and there are only two second-order variables; (Section 5) there is only one second-order variable and it is unary; (Section 6) the following conditions (i–iv) hold for some fixed integer n: (i) the arguments of all second-order variables are ground terms of size $< n$, (ii) the arity of all second-order variables is $< n$, (iii) the number of occurrences of second-order variables is ≤ 5, (iv) there is either a single second-order variable, or there are two second-order variables and no first-order variables.

2 Preliminary Definitions

We assume that the reader is familiar with first-order unification problems, typed λ-calculus [e.g. Hindley & Seldin 1986], and basic notions in term rewriting [e.g. Baader & Nipkow 1998]. We briefly go through the necessary definitions.

A *type* is an expression generated by the grammar $\tau ::= B \mid \tau \to \tau \mid \tau \times \tau$, where B is a set of base types. An *order* of a type τ is defined as follows: $\mathrm{ord}(\tau) = 1$ for any base type $\tau \in B$; $\mathrm{ord}(\tau_1 \times \tau_2) = \max(\mathrm{ord}(\tau_1), \mathrm{ord}(\tau_2))$ and $\mathrm{ord}(\tau_1 \to \tau_2) = \max(\mathrm{ord}(\tau_1)+1, \mathrm{ord}(\tau_2))$ for any composed type. A *signature* is a pair $\langle \Sigma\,,\, \mathcal{X} \rangle$ where Σ is a finite set of *function symbols* and $\mathcal{X}$ is a denumerable set of *variables*. We use ν to denote an arbitrary symbol in a signature. We assume that each function symbol or variable ν of the signature has a type τ, denoted by $\nu : \tau$. In general, the type of a function symbol or a variable is not explicitly given and can be inferred from the context. If $\mathrm{ord}(\tau) = n$ and $\nu : \tau$, then ν is of *order* n. If $\nu : \tau_1 \times \cdots \times \tau_n \to \tau$ then ν has *arity* n or is *n-ary*, denoted $\mathrm{arity}(\nu) = n$. If the type of ν is a base type then $\mathrm{arity}(\nu) = 0$.

In the sequel, all variables are either *first-order* $(x, y, z, X, Y, Z, \dots)$, i.e., of order 1, or *second-order* $(F, G, \dots)$, i.e., of order 2. Similarly, all function symbols are either *first-order* $(a, b, c, \dots)$, also called *constants*, or *second-order* $(f, g, h, \dots)$. For clarity, we use $(x, y, z, \dots)$ for *bound* variables and $(X, Y, Z, \dots)$ for *free* variables.

In second-order typed λ-calculus one can also have function symbols of order greater than 2, but we will avoid them because they do not play an important role in the undecidability results. The same restriction is used by Goldfarb

[1981]. We also assume that we have a *single unique* base type o. Furthermore, we only need terms that are in *β-reduced η-extended normal form* and *de-curried*, i.e., having type $(o\times\cdots\times o)\to o$. With these restrictions *arity* and *type* uniquely determine each other, allowing us to just use the notion of arity. Moreover, any term of type o, in its $\beta\eta$-normal form, is λ-free, i.e., can be written without using lambdas; and, any other term of type $(o\times\cdots\times o)\to o$ can be written using lambdas only as the most external constructor. Thus, we define $\beta\eta$-normal *terms* $(r, l, s, t, u, v, w, \dots)$ to be expressions of the form

$$\lambda x_1 \cdots x_m \,.\, \nu(t_1, \dots, t_n),$$

call it t, where $t_1, \dots, t_n$ have type o, i.e., are *λ-free* terms, and ν is an n-ary symbol from the signature. The term t is said to be of *arity* m. Occasionally, we write $\lambda x \,.\, \lambda y \,.\, t$ for $\lambda xy \,.\, t$. A *first-order* term is a term of type o (a λ-free term) that contains no second-order variables. A term is *closed* if it contains no free variables. A term is *ground* if it contains neither free nor bound variables, i.e., a ground term is a closed first-order term. In some cases, we write terms that are not in $\beta\eta$-normal form. Readers more familiar with first-order notation may interpret them as a meta-notation of their corresponding normal form. In particular, given a term $t = \lambda z_1 \cdots z_m \,.\, u$ (where $z_1, \dots, z_m$ may occur free in u) and terms $t_1, \dots, t_m$, the application term $t(t_1, \dots, t_m)$ denotes its β-reduced form $u\theta$ where $\theta = \{z_i \mapsto t_i \,:\, 1 \le i \le m\}$.

In this paper we only consider *signatures with at least one constant*. In particular, this means that the set of ground terms is always nonempty. For most results this restriction can be relaxed at the expense of having to treat tedious special cases [cf Farmer 1991].

A *substitution* $(\sigma, \theta, \rho \dots)$ is a finite sets of (variable,term) pairs $\{\nu_i \mapsto t_i \,:\, 1 \le i \le n\}$, say θ, where ν_i and t_i have the same arity. The *domain* of θ is $\mathrm{Dom}(\theta) = \{\nu_1, \dots, \nu_n\}$, and the *range* of θ is $\{t_1, \dots, t_n\}$. A substitution is *closed* (*ground*) if its range is a set of closed (ground) terms. Note that the domain of a ground substitution is a set of first-order variables. We will only apply substitutions θ to terms of type o, and we will assume that they are previously $\beta\eta$-normalized. So, the definition is as follows:

$$\begin{array}{ll}
\theta(\nu) = \nu, & \text{if } \nu \notin \mathrm{Dom}(\theta); \\
\theta(\nu) = t, & \text{if } \nu \mapsto t \in \theta; \\
\theta(\nu(t_1, \dots, t_n)) = \nu(\theta(t_1), \dots, \theta(t_n)), & \text{if } \nu \notin \mathrm{Dom}(\theta); \\
\theta(F(t_1, \dots, t_n)) = \rho(u), & \text{if } F \mapsto \lambda x_1 \,.\, \cdots \,.\, \lambda x_n \,.\, u \in \theta; \\
 & \text{and } \rho = \{x_i \mapsto \theta(t_i) \,:\, 1 \le i \le n\}.
\end{array}$$

For better readability we occasionally use the convention $t\theta$ for $\theta(t)$ (or $\nu\theta$ for $\theta(\nu)$). The composition $\theta\circ\rho$ of two substitutions θ and ρ is defined in a standard manner to denote a substitution σ such that $\sigma(t) = \theta(\rho(t))$, for all λ-free terms t.

2.1 Second-order Unification

A *second-order equation* is a pair $t \stackrel{?}{=} u$ of λ-free terms. *Second-order unification* or *SOU* is the problem, given a *system* (finite set) of second-order equations

$\{t_1 \stackrel{?}{=} u_1, \ldots, t_n \stackrel{?}{=} u_n\}$, to decide whether there exists a substitution θ such that $\theta(t_i) = \theta(u_i)$ for all i. Such a θ is said to *unify* or *solve* $\mathcal{S}$. We say that a SOU problem is *simple* if all arguments of all second-order variables are ground terms. Pietrzykowski [1973] was the first to describe a complete semi-decision procedure for second-order unification.

2.2 Simultaneous rigid reachability

Rigid reachability is the problem, given a system R of first-order rules $l \to r$ and two first-order terms s and t, whether there exists a ground substitution θ such that $s\theta$, $t\theta$ and $R\theta$ are ground, and $s\theta$ rewrites in 0 or more steps via $R\theta$ into $t\theta$, i.e., $s\theta \stackrel{*}{\to}_{R\theta} t\theta$.

The expression $R \vdash^{\mathrm{r}} s \to t$ is called a *reachability constraints*, and R its *left-hand side* or *rule set*. The term "rigid" stems from the fact that for no rule more than one instance can be used in the rewriting process. Simultaneous rigid reachability is the problem in which a substitution is sought which simultaneously solves each member of a system (finite set) of reachability constraints. A special case of (simultaneous) rigid reachability arises when the left-hand sides are *symmetric*: containing for each rule $l \to r$ also its converse $r \to l$, in which case the rules are written as equations $l \approx r$. In the symmetric case reachability constraints are also called *rigid equations*. The symmetric case was introduced by Gallier et al. [1987] as *simultaneous rigid E-unification* or *SREU*. (Symmetric systems arise, for instance, from orienting a given set of equations in both directions.) It has been shown by Degtyarev & Voronkov [1995] that SREU is undecidable, whereas the non-simultaneous case of SREU (with just one rigid equation to solve) is NP-complete [Gallier, Narendran, Plaisted & Snyder 1988]. However, rigid reachability is undecidable [Ganzinger, Jacquemard & Veanes 1998].

3 From rigid reachability to SOU

In this section we prove two lemmas that show a close relationship between rigid reachability and second-order unification. These lemmas are used as basic tools in the rest of the paper. Some connections between rigid reachability and SOU were first studied and used by Farmer [1991, Lemma 5.2]. Basically, the main idea is based on a close correspondence between second-order unification problems on a certain form, and ground rewriting. Consider two ground terms s and t, and a system of n ground rules $l_1 \to r_1, \ldots, l_n \to r_n$. Then

$$s \stackrel{*}{\to}_{\{l_1 \to r_1, \ldots, l_n \to r_n\}} t$$

if and only if this second-order equation is unifiable (let f and c be new symbols):

$$F(l_1, \ldots, l_n, f(t, c)) \stackrel{?}{=} f(s, F(r_1, \ldots, r_n, c)).$$

Any unifier θ of this equation maps F to a λ-term $\lambda x_1 \cdots x_n\, y\,.\, u$, where u is a first-order term over the extended signature that encodes a possible reduction

of s to t in the given rewrite system. To see how this works we consider an example.

Example 1 Consider a rewrite system R consisting of two rules $h(a) \to a$ and $g(a) \to a$. Let s be the term $h(g(h(a)))$ and t be the constant a. Then s rewrites in R to t as follows:

$$s \to_{h(a)\to a} h(g(a)) \to_{g(a)\to a} h(a) \to_{h(a)\to a} t.$$

Consider the equation

$$F(h(a), g(a), f(t, c)) \stackrel{?}{=} f(s, F(a, a, c)).$$

A unifier of this equation that encodes the above reduction is illustrated in Figure 1. □

In the following two lemmas, we consider a fixed signature $\langle \Sigma\,,\, \mathcal{X} \rangle$, a constant $c \notin \Sigma$ and a binary function symbol $f \notin \Sigma$. Moreover, without loss of generality, we only consider closed substitutions.

Lemma 1 *Let $l_1, \ldots, l_m, r_1, \ldots, r_m, s, t$ be terms over Σ and θ a substitution such that $l_1\theta, \ldots, l_m\theta, t\theta$ are ground terms over Σ.*

Let $\vec{z}$ denote $z_1, \ldots, z_m$, $\vec{l\theta}$ denote $l_1\theta, \ldots, l_m\theta$ and $\vec{r\theta}$ denote $r_1\theta, \ldots, r_m\theta$. The following statements are equivalent.

(i) θ solves $F(l_1, \ldots, l_m, f(t, c)) \stackrel{?}{=} f(s, F(r_1, \ldots, r_m, c))$

(ii) There exists $k \geq 0$ and closed terms s_i of arity m over Σ for $1 \leq i \leq k$, such that

$$F\theta = \lambda\vec{z}\,.\,\lambda z_{m+1}\,.\,f(s_1(\vec{z}), f(s_2(\vec{z}), \ldots, f(s_k(\vec{z}), z_{m+1})\cdots)), \qquad (1)$$

and, either $k = 0$ and $s\theta = t\theta$, or

$$\begin{aligned} s\theta &= s_1(\vec{l\theta}), && (2)\\ s_i(\vec{r\theta}) &= s_{i+1}(\vec{l\theta}), \quad \text{for } 1 \leq i < k, && (3)\\ s_k(\vec{r\theta}) &= t\theta. && (4) \end{aligned}$$

Proof: **(i)⇐(ii)** It is easy to check that θ satisfying (ii) solves the second-order equation in (i).
(i)⇒(ii) Let θ satisfying (i) be given. Let $F\theta = t_F$. So, by (i),

$$t_F(\vec{l\theta}, f(t\theta, c)) = f(s\theta, t_F(\vec{r\theta}, c)). \qquad (5)$$

We say here that a second-order term u is a *list* if either $u = \lambda\vec{z}\,.\,\lambda z_{m+1}\,.\,z_{m+1}$, or $u = \lambda\vec{z}\,.\,\lambda z_{m+1}\,.\,f(u_1(\vec{z}, z_{m+1}), u_2(\vec{z}, z_{m+1}))$ for some closed terms u_1 and u_2, where u_2 is a list. The following property:

(*) *Let t_1 be a closed term of arity $m+1$, $\vec{t_2}, t_3$ be a sequence of $m+1$ ground terms over Σ, $t_4, \vec{t_5}$ be a sequence of $m+1$ ground terms. If*

$$t_1(\vec{t_2}, f(t_3, c)) = f(t_4, t_1(\vec{t_5}, c))$$

then t_1 is a list.

is proved easily, by induction on the size of t_1, using that $\vec{t_2}, t_3$ is a sequence of ground terms over Σ and that $c, f \notin \Sigma$. The base case is t_1 being $\lambda\vec{z}\,.\,\lambda z_{m+1}\,.\,z_{m+1}$. The induction case is equally straightforward. Alternatively, (*) can be proved using the Pietrzykowski's procedure, and its completeness, and showing that t_1 can only have the form of a list.

From (*) and (5) follows that t_F is a list. Therefore, it has the following form for some $k \geq 0$ and sequence of closed second-order terms s_i, $1 \leq i \leq k$:

$$t_F = \lambda\vec{z}\,.\,\lambda z_{m+1}\,.\,f(s_1(\vec{z}, z_{m+1}), f(\,s_2(\vec{z}, z_{m+1}), \ldots, f(s_k(\vec{z}, z_{m+1}), z_{m+1})\cdots)). \tag{6}$$

If $k = 0$ then (i) implies that $s\theta = t\theta$, and thus (ii) holds.

Assume that $k > 0$. To prove (ii) it remains to be shown that f and c do not occur in any of the s_i's, these s_i's do not use their last argument, i.e., if $s_i = \lambda\vec{z}\,.\,\lambda z_{m+1}\,.\,s_i'$ then z_{m+1} does not occur in s_i', and that (2)–(4) hold.

It follows from (5) and (6) that

$$\begin{array}{llll} f(s_1(\vec{l}\theta, t'), & f(s_2(\vec{l}\theta, t'), & \ldots f(t\theta, & c)\cdots) = \\ f(s\theta, & f(s_1(\vec{r}\theta, c), & \ldots f(s_k(\vec{r}\theta, c), & c)\cdots), \end{array}$$

where $t' = f(t\theta, c)$. Hence, $s_1(\vec{l}\theta, t') = s\theta$, $s_{i+1}(\vec{l}\theta, t') = s_i(\vec{r}\theta, c)$ for $1 \leq i < k$, and $t\theta = s_k(\vec{r}\theta, c)$. It follows by induction on $k - i$ (with $k = i$ being the base case) that each s_i has the required from, by using that all terms in $t\theta, \vec{l}\theta$ are ground terms over Σ and $f, c \notin \Sigma$. The lemma follows. ■

A unifier of a second-order equation of the form in Lemma 1, as well as the possible rewriting sequences for a term, are not unique as the following example shows. Moreover, if we read conditions (2)–(4) as "$s\theta$ rewrites to $t\theta$", then we have to consider parallel rewriting steps.

Example 2 The equation

$$F(a, f(g(b, b), c)) \stackrel{?}{=} f(g(a, a), F(b, c))$$

has three possible solutions:

$$\begin{array}{l} \{F \mapsto \lambda x\,y\,.\,f(g(x, a), f(g(b, x), y))\}, \\ \{F \mapsto \lambda x\,y\,.\,f(g(a, x), f(g(x, b), y))\}, \\ \{F \mapsto \lambda x\,y\,.\,f(g(x, x), y)\}, \end{array}$$

corresponding to the three possible reductions from $g(a,a)$ to $g(b,b)$ that can be done applying the rule $a \to b$ in "parallel":

$$\begin{array}{l} g(a,a) \to g(b,a) \to g(b,b) \\ g(a,a) \to g(a,b) \to g(b,b) \\ g(a,a) \overset{||}{\to} g(b,b) \end{array}$$

In the third case we write $\overset{||}{\to}$ to emphasize that this rewriting step is done in parallel over two disjoint redexes. The way how the first substitution solves the equation is illustrated by Figure 2. □

We can now state our main lemma that links together rigid reachability and second-order unification.

Lemma 2 *Let $R = \{l_i \to r_i : 1 \le i \le m\}$ be a set of rules over Σ and let s and t be terms over Σ. Statements (i) and (ii) are equivalent for all θ such that $t\theta, l_1\theta, \ldots, l_m\theta, r_1\theta, \ldots, r_m\theta$ are ground terms over Σ. Let F be a second-order variable such that $F \notin \mathrm{Dom}(\theta)$.*

(i) $\theta \cup \{F \mapsto t'\}$ solves $F(l_1, \ldots, l_m, f(t,c)) \stackrel{?}{=} f(s, F(r_1, \ldots, r_m, c))$ for some t'.

(ii) $s\theta$ is a ground term over Σ and θ solves $R \vdash^{\mathrm{r}} s \to t$.

Proof: **(i)⇒(ii)** Let $\theta' = \theta \cup \{F \mapsto t'\}$ satisfying (i) be given. There are two cases by Lemma 1((i)⇒(ii)). The first case is that $t' = \lambda\vec{z}\,.\,\lambda z_{m+1}\,.\,z_{m+1}$ and $t\theta' = s\theta'$, and thus $t\theta = s\theta$. Hence, (ii) holds trivially. The second case is that there exists a $k \ge 1$ and closed terms s_i for $1 \le i \le k$, such that

$$t' = \lambda\vec{z}\,.\,\lambda z_{m+1}\,.\,f(s_1(\vec{z}), f(s_2(\vec{z}), \ldots, f(s_k(\vec{z}), z_{m+1})\cdots))$$

and $s\theta = s_1(\vec{l\theta})$, $s_i(\vec{r\theta}) = s_{i+1}(\vec{l\theta})$, for $1 \le i < k$, $s_k(\vec{r\theta}) = t\theta$. Clearly, $s_i(\vec{l\theta}) \stackrel{*}{\to}_{R\theta} s_i(\vec{r\theta})$ for $1 \le i \le k$, and thus $s_i(\vec{l\theta}) \stackrel{*}{\to}_{R\theta} s_{i+1}(\vec{l\theta})$, for $1 \le i < k$. It follows that

$$s\theta = s_1(\vec{l\theta}) \stackrel{*}{\to}_{R\theta} s_k(\vec{l\theta}) \stackrel{*}{\to}_{R\theta} s_k(\vec{r\theta}) = t\theta,$$

as needed.
(ii)⇒(i) Let θ satisfying (ii) be given We have a reduction:

$$s\theta = t_0 \to_{R\theta} t_1 \to_{R\theta} \cdots \to_{R\theta} t_{k-1} \to_{R\theta} t_k = t\theta.$$

If $k = 0$ then let $t' = \lambda\vec{z}\,.\,\lambda z_{m+1}\,.\,z_{m+1}$ and (i) follows from Lemma 1((ii)⇒(i)). Assume that $k \ge 1$ and consider a fixed i, $1 \le i \le k$. The rewrite step $t_{i-1} \to_R t_i$ uses a rule $l_j\theta \to r_j\theta$ for some j, $1 \le j \le m$, and replaces a certain subterm occurrence of $l_j\theta$ in t_{i-1} by $r_j\theta$. Construct s_i from t_{i-1} by replacing that occurrence of $l_j\theta$ by z_j. So $s_i(\vec{l\theta}) = t_{i-1}$ and $s_i(\vec{r\theta}) = t_i$. Given such s_i for $1 \le i \le k$, obviously (2–4) are true. Let

$$t' = \lambda\vec{z}\,.\,\lambda z_{m+1}\,.\,f(s_1(\vec{z}), f(s_2(\vec{z}), \ldots, f(s_k(\vec{z}), z_{m+1})\cdots))$$

and (i) follows from Lemma 1((ii)⇒(i)). ■

The following example shows why the additional restrictions on the signature are crucial in Lemma 2.

Example 3 The reachability constraint

$$\{X \to b\} \vdash^{\mathrm{r}} g(a,d) \to g(b,b)$$

is unsolvable. However, the second-order equation

$$F(X, f(g(b,b),c)) \stackrel{?}{=} f(g(a,d), F(b,c))$$

has a solution θ such that $F\theta = \lambda z_1 \,.\, \lambda z_2 \,.\, z_1$ and $X\theta = f(g(a,d),b)$. Notice that this substitution violates the restriction that $X\theta$ is a ground term over $\{a,b,d,g\}$. □

4 From SREU to SOU

We apply the results in the previous section to show that there is a polynomial time reduction of SREU to SOU. The converse reduction, from SOU to SREU, is given in [Degtyarev & Voronkov 1996] and has also polynomial time complexity. We use a restricted form of SREU that is polynomial time equivalent to SREU and allows us to apply Lemma 2 in a direct manner.

Let $\mathcal{R}$ be a system of reachability constraints over Σ, and Let X be a variable that occurs in $\mathcal{R}$. A *guard for X in $\mathcal{R}$*, if one exists, is any reachability constraint $R \vdash^{\mathrm{r}} s \to t$ in $\mathcal{R}$ such that: R is a set of ground rules; t is a ground term; X occurs in s. The system $\mathcal{R}$ is called *guarded* if there is a guard in $\mathcal{R}$ for each variable that occurs in $\mathcal{R}$.

Guarded simultaneous rigid reachability is the restriction of simultaneous rigid reachability to guarded systems. *Guarded SREU* is the symmetric form of this problem. Guardedness is an important and useful notion also in other contexts [Gurevich & Veanes 1999]. We will use the following fact, making use of a technique due to Degtyarev & Voronkov [1995].

Lemma 3 *SREU is polynomial time equivalent to guarded SREU.*

Proof: For each variable X in the given system of rigid equations (over Σ), add the rigid equation

$$\{f(c,\ldots,c) \approx c \,:\, f \in \Sigma\} \vdash^{\mathrm{r}} X \to c$$

to the system, where c is a constant in Σ. The new system is guarded and is solvable if and only if the original system is solvable [Degtyarev & Voronkov 1995]. ■

The following result constituted a very transparent undecidability proof of SREU.

Theorem 4 (Degtyarev & Voronkov [1996]) *There is a polynomial time reduction of SOU to SREU.*

The converse direction follows from the following theorem and Lemma 3.

Theorem 5 *There is a polynomial time reduction of guarded simultaneous rigid reachability to SOU restricted to equation systems $\mathcal{S}$, where each equation in $\mathcal{S}$ has the form $F(\vec{l}, f(t,c)) \stackrel{?}{=} f(s, F(\vec{r},c))$ with all terms in $s,t,\vec{l},\vec{r}$ being first-order and F occurring only in this equation.*

Proof: Let $\mathcal{R} = \{R_i \vdash^{\mathrm{r}} s_i \to t_i \,:\, 1 \le i \le n\}$ be a guarded system of reachability constraints over a signature Σ. Let c and f be new function symbols with arities 0 and 2, respectively. For each reachability constraints $R_i \vdash^{\mathrm{r}} s_i \to t_i$, $R_i = \{l_j \to r_j \,:\, 1 \le j \le m_i\}$, construct the second-order equation S_i:

$$F_i(l_1, \ldots, l_{m_i}, f(t,c)) \stackrel{?}{=} f(s, F_i(r_1, \ldots, r_{m_i}, c)),$$

where F_i is a new second-order variable. Let $\mathcal{S} = \{S_i \,:\, 1 \le i \le n\}$.

Consider any substitution θ that is undefined for the F_i's. We prove that some extension of θ solves $\mathcal{S}$ if and only if θ solves $\mathcal{R}$.

($\Rightarrow$) Assume that $\theta' = \theta \cup \{F_i \mapsto t'_i \,:\, 1 \le i \le n\}$ solves $\mathcal{S}$.

First, we show that, for $1 \le i \le n$, $R_i\theta$ is a set of ground rules over Σ and $t_i\theta$ is a ground term over Σ. Consider a fixed i. Let X be a variable in R_i or t_i. Let $R_j \vdash^{\mathrm{r}} s_j \to t_j$ be a guard for X in $\mathcal{R}$. So R_j and t_j are ground and $\theta \cup \{F_j \mapsto t'_j\}$ solves S_j. It follows from Lemma 2((i)$\Rightarrow$(ii)) that $s_j\theta$ is a ground term over Σ, and thus, so is $X\theta$.

Second, we apply Lemma 2((i)$\Rightarrow$(ii)) again to all the S_i's to show that θ' solves $\mathcal{R}$.

($\Leftarrow$) By Lemma 2((i)$\Leftarrow$(ii)) and the fact that all the F_i's are distinct. ■

The following example illustrates the importance of guardedness. The proof of Theorem 5 fails already for very simple systems of rigid equations that are not guarded.

Example 4 Consider the system $\mathcal{R} = \{\emptyset \vdash^{\mathrm{r}} X \to Y\}$ where X and Y are variables. Then the corresponding system $\mathcal{S}$ of second-order equations has the form $\{F(f(X,c)) \approx f(Y, F(c))\}$. Let $\theta = \{F \mapsto \lambda z\,.\, f(z,z), X \mapsto c, Y \mapsto f(c,c)\}$. It is easy to check that θ solves $\mathcal{S}$ but it does not solve $\mathcal{R}$. □

We get the following results.

Corollary 6 *The following problems are polynomial time equivalent: SREU; SOU; SOU restricted to equation systems $\mathcal{S}$, where each equation in $\mathcal{S}$ has the form $F(\vec{l}, f(t,c)) \stackrel{?}{=} f(s, F(\vec{r},c))$ with all terms in $s,t,\vec{l},\vec{r}$ being first-order and F occurring only in this equation.*

This shows that already this very restricted form of SOU is undecidable. In contrast, the corresponding *linear* SOU problem is decidable [Levy 1996]. In [1998] the authors proved that guarded simultaneous rigid reachability is

undecidable already when restricted to systems of at most *two* reachability constraints. This result and the proof of Theorem 5 imply the following, even more restrictive, undecidability result.

Corollary 7 *Second-order unification is undecidable under the following restrictions on equation systems: there are at most two distinct second-order variables; every second-order variable occurs at most twice and in at most one equation; every equation contains at most one second-order variable.*

One may wonder if this result can be improved, and indeed it can. Ganzinger et al. [1998] proved recently that rigid reachability is undecidable (already for ground rules). Using this result, the particular structure of the reachability constraints, and an analogue of Lemma 2 (Lemma 2 is not directly applicable to those reachability constraints), they concluded that SOU is undecidable already when there is a single second-order variable that occurs at most twice.

There is an important difference between the reduction from SREU to SOU on one hand and the reduction from SOU to SREU on the other hand. In the former reduction one needs a binary function symbol, whereas the latter reduction shows that monadic SOU reduces to monadic SREU. The use of the binary function symbol in the former reduction seems to be unavoidable because of the following reason. Decidability of monadic second-order unification can be proved by reduction to word equations [Farmer 1988], whereas monadic SREU is only known to reduce to a nontrivial extension of word equations [Gurevich & Voronkov 1997], and its decidability is an open problem.

5 One Second-Order Variable is Enough

The principal result of this section is that the number of *different* second-order variables in SOU plays a minor role compared to the total number of *occurrences* of second-order variables. We present a straightforward reduction of arbitrary systems of second-order equations to systems of second-order equations using just one second-order variable and additional first-order variables. The encoding technique that we use is similar to the techniques used by Farmer [1991]. Several important properties are preserved by that reduction: *maximal arity of variables*, *number of occurrences of second-order variables* and *simplicity*. Let $\mathcal{S}$ be a system of second-order equations. We write $\mathrm{occ}^2(\mathcal{S})$ for the number of occurrences of *second-order variables* in $\mathcal{S}$. We write $\mathrm{arity}^2(\mathcal{S})$ for the maximal arity of the second-order variables in $\mathcal{S}$.

An *interpolation equation* is a second-order equation of the form $F(\vec{t}) \stackrel{?}{=} s$, where F is a second-order variable, $\vec{t}$ is a sequence of first-order terms, and s is a first-order term. The following fact is well-known and easy to prove [e.g. Degtyarev & Voronkov 1996].

Lemma 8 *Let $\mathcal{S}$ be a system of second-order equations. There is a system $\mathcal{S}'$ of interpolation equations such that $\mathcal{S}'$ is solvable if and only if $\mathcal{S}$ is solvable. Moreover,* $\mathrm{arity}^2(\mathcal{S}) = \mathrm{arity}^2(\mathcal{S}')$, $\mathrm{occ}^2(\mathcal{S}) = \mathrm{occ}^2(\mathcal{S}')$, *and if $\mathcal{S}$ is simple then so is $\mathcal{S}'$.*

We now define a reduction from a system of interpolation equations to a system of interpolation equations that uses just one second-order variable with arity equal to the maximal arity of variables in the original system. Let $\mathcal{S}$ be a system of interpolation equations:

$$\bigcup_{1\leq i\leq m} \{F_i(\vec{s}_{ij}) \stackrel{?}{=} t_{ij} \ : \ 1 \leq j \leq k_i\}$$

where $\{F_1, \ldots, F_m\}$ are the different second-order variables in $\mathcal{S}$, and for each F_i there are k_i interpolation equations. Assume, without loss of generality, that the arities of the F_i's are equal. Otherwise, if some F_i has arity less than $\text{arity}^2(\mathcal{S})$ then we can increase the arity of F_i to $\text{arity}^2(\mathcal{S})$ by replacing each occurrence $F_i(\vec{s}_{ij})$ by $F_i(\vec{s}_{ij}, s, \ldots, s)$, where s is the last term in the sequence $\vec{s}_{ij}$. Clearly, this neither affects the solvability of $\mathcal{S}$, nor the other properties mentioned above.

Let G be a second-order variable with arity equal to $\text{arity}^2(\mathcal{S})$, let g be a new m-ary function symbol, and let $\mathcal{S}'$ denote the system of second-order equations consisting of

$$\bigcup_{1\leq i\leq m} \{G(\vec{s}_{ij}) \stackrel{?}{=} g(\underbrace{\_, \ldots, \_}_{i-1}, t_{ij}, \underbrace{\_, \ldots, \_}_{m-i}) \ : \ 1 \leq j \leq k_i\}$$

and, unless all elements of some sequence $\vec{s}_{ij}$ are not variables[1], the equation

$$G(c, c, \ldots, c) \stackrel{?}{=} g(\_, \_, \ldots, \_),$$

where c is a constant and each occurrence of '$\_$' denotes a distinct first-order variable. By combining Lemma 8 with this reduction we can prove the following result.

Theorem 9 *Let $\mathcal{S}$ be a system of second-order equations. There is a system of interpolation equations $\mathcal{S}'$, with at most one second-order variable, such that $\mathcal{S}'$ is solvable if and only if $\mathcal{S}$ is solvable.*
Moreover, $\text{arity}^2(\mathcal{S}) = \text{arity}^2(\mathcal{S}')$, $\text{occ}^2(\mathcal{S}) \leq \text{occ}^2(\mathcal{S}') \leq \text{occ}^2(\mathcal{S}) + 1$, *and if $\mathcal{S}$ is simple then so is $\mathcal{S}'$.*

Proof: Assume, without loss of generality, that $\mathcal{S}$ consists solely of interpolation equations. We prove that $\mathcal{S}$ is solvable if and only if $\mathcal{S}'$ is solvable. We just consider a special case. The proof of the general case is analogous. Let $\mathcal{S}$ be the following system:

$$\begin{array}{lcl} F_1(\vec{s}_1) & \stackrel{?}{=} & t_1 \\ F_2(\vec{s}_2) & \stackrel{?}{=} & t_2 \\ F_3(\vec{s}_3) & \stackrel{?}{=} & t_3 \end{array}$$

[1] This condition can be weakened.

Then $\mathcal{S}'$ is the following system:

$$\begin{array}{rcl} G(\vec{s}_1) & \stackrel{?}{=} & g(t_1, X_{12}, X_{13}) \\ G(\vec{s}_2) & \stackrel{?}{=} & g(X_{21}, t_2, X_{23}) \\ G(\vec{s}_3) & \stackrel{?}{=} & g(X_{31}, X_{32}, t_3) \\ (\quad G(\vec{c}) & \stackrel{?}{=} & g(X_1, X_2, X_3) \quad) \end{array}$$

(⇒) Assume θ solves $\mathcal{S}$. Define θ' as follows. For all first-order variables in $\mathcal{S}$, θ' agrees with θ. For G, $\theta'(G) = \lambda\vec{x}\,.\, g(\theta(F_1(\vec{x})), \theta(F_2(\vec{x})), \theta(F_3(\vec{x})))$. For the new first-order variables $\{X_{12}, X_{13}, X_{21}, X_{23}, X_{31}, X_{32}\}$, let $\theta'(X_{ij}) = \theta(F_j(\vec{s}_i))$. For $\{X_1, X_2, X_3\}$, let $\theta'(X_j) = \theta(F_j(\vec{c}))$.

Consider the first equation of $\mathcal{S}'$. We have that

$$\begin{array}{rcl} \theta'(G(\vec{s}_1)) & = & g(\theta(F_1(\vec{s}_1)), \theta(F_2(\vec{s}_1)), \theta(F_3(\vec{s}_1))) \\ & = & g(\theta(t_1), \theta'(X_{12}), \theta'(X_{13})) \\ & = & \theta'(g(t_1, X_{12}, X_{13})) \end{array}$$

Hence, θ' solves $G(\vec{s}_1) \stackrel{?}{=} g(t_1, X_{12}, X_{13})$. The other cases are similar.

(⇐) Assume that θ' solves $\mathcal{S}'$, we construct a substitution θ that solves $\mathcal{S}$. We have $\theta'(G) = \lambda\vec{x}\,.\, g(u_1(\vec{x}), u_2(\vec{x}), u_3(\vec{x}))$ for some closed terms u_1, u_2 and u_3, or else, θ' would not solve either an equation in $\mathcal{S}'$ where all arguments of G are non-variables (note that g does not occur in those arguments), or the optional equation $G(\vec{c}) \stackrel{?}{=} g(X_1, X_2, X_3)$.

Define θ so that it agrees with θ' on first-order variables in $\mathcal{S}$ and $\theta(F_i) = u_i$ for $1 \leq i \leq 3$. By applying θ' to the left-hand side of the first equation of $\mathcal{S}'$ we have that

$$\begin{array}{rcl} \theta'(G(\vec{s}_1)) & = & g(u_1(\theta'(\vec{s}_1)), u_2(\theta'(\vec{s}_1)), u_3(\theta'(\vec{s}_1))) \\ & = & g(\theta(F_1(\vec{s}_1)), u_2(\theta'(\vec{s}_1)), u_3(\theta'(\vec{s}_1))), \end{array}$$

and by applying θ' to the right-hand side of the same equation we have that

$$\theta'(g(t_1, X_{12}, X_{13})) = g(\theta(t_1), \theta'(X_{12}), \theta'(X_{13}).$$

But θ' solves $\mathcal{S}'$, and thus $\theta(F_1(\vec{s}_1)) = \theta(t_1)$. Hence, θ solves $F_1(\vec{s}_1) \stackrel{?}{=} t_1$. The other cases are similar. ■

The above reduction implies, by using Goldfarb's [1981] result, the undecidability of SOU with a single second-order variable. By using the following theorem, we get a stronger result.

Theorem 10 (Farmer [1991]) *Second-order unification is undecidable even when all second-order variables are unary.*

Farmer's theorem and Theorem 9 implies the following.

Corollary 11 *Second-order unification is undecidable already with a single unary second-order variable.*

This result may be sharpened, by using Farmer's [1991] encoding techniques, to hold already when the only function symbols in the signature are a constant c and a binary function symbol f.

6 Undecidability of SOU revisited

We present an elementary undecidability proof of SOU, by a direct encoding from the halting problem for Turing machines. The proof by Goldfarb [1981] uses Hilbert's tenth problem. The latter problem is proved undecidable by Matiyasevich [1970], by using sophisticated number-theoretical methods. We use techniques from [Plaisted 1995, Gurevich & Veanes 1999], and apply them in the context of SOU, by using the lemmas in Section 3.

We consider a fixed deterministic Turing machine M with *initial state* q_0, *final state* q_f, a *blank* character $\sqcup$, and an *input alphabet* that does not include the blank. The tape of M is infinite in only one direction. By Σ_M we denote the set of all the symbols in M, i.e., the states, the input characters and the blank. All elements of Σ_M are assigned arity 0, i.e., are treated as constants. We assume, without loss of generality, that before acceptance M erases its tape contents, i.e., M is only allowed to write a blank when it erases the *last* non-blank symbol on the tape and continues to erase all the symbols before it, before entering the final state with the empty tape.

An *ID* of M is any string vqw where vw is a string over the input alphabet of M and q is a state of M. In particular, the *initial* ID of M for input string v has the form q_0v, and the *final* ID is simply the one character string q_f. A *move* of M is any pair of strings (v, v^+) where v is an ID and v^+ is the successor of v according to the transition function of M, if v is non-final. The successor of the final state is the empty string (ϵ), i.e., $q_\mathrm{f}^+ = \epsilon$.

6.1 Main Idea

We construct two second-order equations $S^M_\mathrm{mv}(F, G)$ and $S^M_\mathrm{sp}(X, F)$ from M, that have roughly the following properties: for any substitution θ and initial ID q_0v_0 that is represented by a term t_0,

1. θ solves $S^M_\mathrm{mv}(F, G)$ (for some $G\theta$) if and only if $F\theta$ represents a sequence of moves of M:
$$((v_1, v_1^+), (v_2, v_2^+), \ldots, (v_k, v_k^+)).$$
2. θ solves $S^M_\mathrm{sp}(t_0, F)$ if and only if $F\theta$ represents the *shifted pairing* of a sequence $(v_1, v_2, \ldots, v_k)$ of IDs of M, where $v_1 = q_0v_0$ and $v_k = q_\mathrm{f}$. (See Figure 3.)

Consequently, θ solves both second-order equations if and only if $F\theta$ represents (the shifted pairing of) the valid computation of M with input v_0.

6.2 Encoding Sequences of Moves

We introduce a family of new constants $\{c_{ab} \mid a, b \in \Sigma_M\}$ and use them to encode moves of M in the following manner. Let $v = a_1 a_2 \cdots a_m$ be any ID of M and let $v^+ = b_1 b_2 \cdots b_n$ be its successor. Note that $m - 1 \leq n \leq m + 1$. We let $\langle v\,,\, v^+\rangle$ denote the following string:

$$\langle v\,,\, v^+\rangle = \begin{cases} c_{a_1 b_1} c_{a_2 b_2} \cdots c_{a_m b_m} c_{\sqcup b_n}, & \text{if } n = m+1; \\ c_{a_1 b_1} c_{a_2 b_2} \cdots c_{a_n b_n} c_{a_m \sqcup}, & \text{if } n = m-1; \\ c_{a_1 b_1} c_{a_2 b_2} \cdots c_{a_m b_n}, & \text{if } n = m. \end{cases}$$

we call such a string a *move* of M or an *M-move*. Note that $\langle q_{\mathrm{f}}\,,\, \epsilon\rangle = c_{q_{\mathrm{f}}\sqcup}$. Intuitively, a blank is added at the end of the shorter of the two strings (in case they differ in length) and the pair of the resulting strings is encoded character by character.

We fix two new constants c_{w} and c_{t} and two new binary function symbols f_{w} and f_{t}. Let Σ_{id} and Σ_{mv} be the following signatures:

$$\Sigma_{\mathrm{id}} = \Sigma_M \cup \{c_{\mathrm{w}}, f_{\mathrm{w}}\}$$
$$\Sigma_{\mathrm{mv}} = \{c_{ab} \mid a, b \in \Sigma_M\} \cup \{c_{\mathrm{w}}, f_{\mathrm{w}}, c_{\mathrm{t}}, f_{\mathrm{t}}\}$$

A term s is a called a *word* if either $s = c_{\mathrm{w}}$ (the *empty word*), or $s = f_{\mathrm{w}}(c, s')$ for some constant c that is distinct from c_{w} and word s'. Whenever convenient, we write a word as follows:

$$f_{\mathrm{w}}(a_1, f_{\mathrm{w}}(a_2, \cdots f_{\mathrm{w}}(a_n, c_{\mathrm{w}}) \cdots)) = \text{`}a_1 a_2 \cdots a_n\text{'}$$

and say that the word *represents* the string. A term t is called a *train*, if either $t = c_{\mathrm{t}}$ (the *empty train*), or $t = f_{\mathrm{t}}(s, t')$ for some word s and train t'. So trains are simply representations of string sequences. Conceptually we identify words with strings and trains with sequences of strings.

A train that represents a sequence of M-moves is called a *move-train* of M or an *M-move-train*.

Example 5 Consider a Turing machine M that in state q, overwrites any symbol by $\sharp$, moves right and remains in state q. The following term is a move-train of M:

$$f_{\mathrm{t}}(\text{`}\langle q\,,\, \sharp q\rangle\text{'}, f_{\mathrm{t}}(\text{`}\langle aqaa\,,\, a\sharp qa\rangle\text{'}, f_{\mathrm{t}}(\text{`}\langle \sharp q\,,\, \sharp\sharp q\rangle\text{'}, c_{\mathrm{t}})))$$

Note that two consecutive moves in move-trains need not be related. □

Lemma 12 follows from [Gurevich & Veanes 1999, Train Lemma] and the fact that the set of all moves of M is regular set of strings. (See also [Gurevich & Veanes 1999, Lemma 18].) It is used together with Lemma 2 to construct the second-order equation $S^M_{\mathrm{mv}}(F, G)$ with the desired properties.

Lemma 12 *There is a ground rewrite system R_{mv} over $\Sigma_{\mathrm{mv}} \cup Q_{\mathrm{mv}}$, where Q_{mv} is a set of constants disjoint from Σ_{mv}, with rules of the form $f(a, b) \to c$ and $a \to b$, such that, for all ground terms t over Σ_{mv}, t is a move-train of M if and only if $t \xrightarrow{*}_{R_{\mathrm{mv}}} c_{\mathrm{t}}$.*

6.3 The Main Reduction

Let R_{mv} and Q_{mv} be given by Lemma 12, such that $\Sigma_{\mathrm{mv}} \cap Q_{\mathrm{mv}} = \emptyset$. Let $m = |\Sigma_M|^2$ and let $((a_i, b_i))_{1 \le i \le m}$ be a fixed sequence of all the pairs of constants from Σ_M. We will be using the following definitions and shorthand notations, throughout the rest of this section.

- $\vec{a}$ denotes the sequence $a_1, a_2, \ldots, a_m$ of constants.
- $\vec{b}$ denotes the sequence $b_1, b_2, \ldots, b_m$ of constants.
- $\vec{c}$ denotes the sequence $c_{a_1 b_1}, c_{a_2 b_2}, \ldots, c_{a_m b_m}$ of constants.
- $\vec{z}$ denotes the sequence $z_{a_1 b_1}, z_{a_2 b_2}, \ldots, z_{a_m b_m}$ of distinct variables; we also write z_i for $z_{a_i b_i}$.
- Let $R_{\mathrm{mv}} = \{l_i \to r_i : 1 \le i \le |R_{\mathrm{mv}}|\}$, let $\vec{l}_{\mathrm{mv}}$ denote the sequence $l_1, \ldots, l_{|R_{\mathrm{mv}}|}$, and let $\vec{r}_{\mathrm{mv}}$ denote the sequence $r_1, \ldots, r_{|R_{\mathrm{mv}}|}$.
- d is a new constant and g is a new binary function symbol.

For any ground term t, we define the second-order equations $S^M_{\mathrm{sp}}(t, F)$ and $S^M_{\mathrm{mv}}(F, G)$ as follows:

$$S^M_{\mathrm{sp}}(t, F): \quad F(\vec{a}, \text{`}\sqcup\text{'}, c_{\mathrm{w}}, f_{\mathrm{t}}(\text{`}\sqcup\text{'}, c_{\mathrm{t}})) \stackrel{?}{=} f_{\mathrm{t}}(t, F(\vec{b}, c_{\mathrm{w}}, \text{`}\sqcup\text{'}, c_{\mathrm{t}}))$$

$$S^M_{\mathrm{mv}}(F, G): \quad G(\vec{l}_{\mathrm{mv}}, g(c_{\mathrm{t}}, d)) \stackrel{?}{=} g(F(\vec{c}, c_{\mathrm{w}}, c_{\mathrm{w}}, c_{\mathrm{t}}), G(\vec{r}_{\mathrm{mv}}, d))$$

Let us briefly recall the intuition behind this construction. Assume that θ solves this system of two equations. First, consider $S^M_{\mathrm{mv}}(F\theta, G\theta)$. It follows from Lemma 12 (with a little help from Lemma 13 below) that $\theta(F(\vec{c}, c_{\mathrm{w}}, c_{\mathrm{w}}, c_{\mathrm{t}}))$ is a ground term over Σ_{mv} representing a sequence of moves of M:

$$(\langle v_1\ ,\ v_1^+\rangle, \langle v_2\ ,\ v_2^+\rangle, \ldots, \langle v_k\ ,\ v_k^+\rangle)$$

Second, consider $S^M_{\mathrm{sp}}(t, F\theta)$ and Lemma 1. This tells us that $F\theta$ is a term over the signature $\Sigma_{\mathrm{id}} \cup \{f_{\mathrm{t}}, c_{\mathrm{t}}\}$, and thus, cannot contain constants from $\vec{c}$. Consequently, without loss of generality, $F\theta$ has the form $\lambda\vec{z}\lambda x_1\, x_2\, y.t_F$, where every occurrence of c_{ab} in $\theta(F(\vec{c}, c_{\mathrm{w}}, c_{\mathrm{w}}, c_{\mathrm{t}}))$ corresponds to z_{ab} in t_F. Therefore $\theta(F(\vec{a}, \text{`}\sqcup\text{'}, c_{\mathrm{w}}, f_{\mathrm{t}}(\text{`}\sqcup\text{'}, c_{\mathrm{t}})))$ represents (roughly) the sequence:

$$(v_1, v_2, \ldots, v_k, \sqcup)$$

and $f_{\mathrm{t}}(\text{`ID}_0\text{'}, \theta(F(\vec{b}, c_{\mathrm{w}}, \text{`}\sqcup\text{'}, c_{\mathrm{t}})))$, where $\text{`ID}_0\text{'} = t$, represents (roughly) the sequence:

$$(\mathrm{ID}_0, v_1^+, v_2^+, \ldots, v_k^+)$$

But these two sequences must be equal, which implies that $(v_1, v_2, \ldots, v_k)$ is a valid computation of M.

Example 6 Suppose we have a Turing machine M that accepts the one-character sequence "a". A valid computation of M can have the form

$$(q_0a, aq_1, q_2a, q_\mathrm{f}).$$

A solution $F\theta$ of the second-order system encodes this valid computation of M as is illustrated in Figure 4. □

Before proving the main theorem we prove a simple lemma.

Lemma 13 *Given a ground term t over Σ_id and a substitution θ, if θ solves $\{S^M_\mathrm{sp}(t,F), S^M_\mathrm{mv}(F,G)\}$ then $F\theta$ is a closed term of arity $m+3$ over $\{c_\mathrm{w}, f_\mathrm{w}, c_\mathrm{t}, f_\mathrm{t}\}$.*

Proof: Let t and θ be given, and assume that θ solves $\{S^M_\mathrm{sp}(t,F)\,,\, S^M_\mathrm{mv}(F,G)\}$. It follows from θ solving $S^M_\mathrm{sp}(t,F)$ and Lemma 1 that $F\theta$ is a closed term of arity $m+3$ over the signature $\Sigma_\mathrm{id}\cup\{c_\mathrm{t}, f_\mathrm{t}\}$. It follows from θ solving $S^M_\mathrm{mv}(F,G)$ and again Lemma 1 that $\theta(F(\vec{c}, c_\mathrm{w}, c_\mathrm{w}, c_\mathrm{t})) = s_1(\vec{l}_\mathrm{mv})$ for some closed term s_1 over $\Sigma_\mathrm{mv}\cup Q_\mathrm{mv}$. Hence, $\theta(F(\vec{c}, c_\mathrm{w}, c_\mathrm{w}, c_\mathrm{t}))$ is a ground term over $\Sigma_\mathrm{mv}\cup Q_\mathrm{mv}$. So $F\theta$ is a closed term of arity $m+1$ over $(\Sigma_\mathrm{mv}\cup Q_\mathrm{mv})\cap(\Sigma_\mathrm{id}\cup\{c_\mathrm{t}, f_\mathrm{t}\}) = \{c_\mathrm{w}, f_\mathrm{w}, c_\mathrm{t}, f_\mathrm{t}\}$. ■

We can now prove the main theorem. Recall that the initial state of M is q_0.

Theorem 14 *For any input string v_0 for M, the second-order unification problem $\{S^M_\mathrm{sp}(\text{`}q_0v_0\text{'}, F)\,,\, S^M_\mathrm{mv}(F,G)\}$ is solvable if and only if M accepts v_0.*

Proof: Let v_0 be given.

(⇒) Assume that the system $\{S^M_\mathrm{sp}(\text{`}q_0v_0\text{'}, F)\,,\, S^M_\mathrm{mv}(F,G)\}$ is solvable, and let θ be a solution. Since θ solves $S^M_\mathrm{mv}(F,G)$, it follows from Lemma 2 that

$$\theta(F(\vec{c}, c_\mathrm{w}, c_\mathrm{w}, c_\mathrm{t})) \xrightarrow{*}_{R_\mathrm{mv}} c_\mathrm{t},$$

and it follows from Lemma 13 that $\theta(F(\vec{c}, c_\mathrm{w}, c_\mathrm{w}, c_\mathrm{t}))$ is a ground term over Σ_mv. Hence, by Lemma 12, $\theta(F(\vec{c}, c_\mathrm{w}, c_\mathrm{w}, c_\mathrm{t}))$ is a move-train of M:

$$\theta(F(\vec{c}, c_\mathrm{w}, c_\mathrm{w}, c_\mathrm{t})) = f_\mathrm{t}(\text{`}\langle v_1\,,\, v_1^+\rangle\text{'}, \cdots f_\mathrm{t}(\text{`}\langle v_k\,,\, v_k^+\rangle\text{'}, c_\mathrm{t})\cdots), \tag{7}$$

where each v_i is an ID of M and $k \geq 0$. But θ solves also $S^M_\mathrm{sp}(\text{`}q_0v_0\text{'}, F)$, hence, it follows from Lemma 1 that

$$\theta(F) = \lambda\vec{z}\,.\,\lambda x_1\,x_2\,y\,.\,f_\mathrm{t}(s_1(\vec{z}, x_1, x_2), \cdots f_\mathrm{t}(s_{k'}(\vec{z}, x_1, x_2), y)\cdots), \tag{8}$$

where

$$\text{`}q_0v_0\text{'} = s_1(\vec{a}, \text{`}\sqcup\text{'}, c_\mathrm{w}), \tag{9}$$

$$s_i(\vec{b}, c_\mathrm{w}, \text{`}\sqcup\text{'}) = s_{i+1}(\vec{a}, \text{`}\sqcup\text{'}, c_\mathrm{w}), \quad \text{for } 1 \leq i < k, \tag{10}$$

$$s_{k'}(\vec{b}, c_\mathrm{w}, \text{`}\sqcup\text{'}) = \text{`}\sqcup\text{'}. \tag{11}$$

The case $k' = 0$, i.e., $\theta(F) = \lambda\vec{z}\,.\,\lambda x_1\,x_2\,y\,.\,y$, is not possible because 'q_0v_0' $\neq$ '$\sqcup$'. By comparing (7) and (8), and using Lemma 13, every $s_i(\vec{z}, x_1, x_2)$ is a term over $\{c_{\mathrm{w}}, f_{\mathrm{w}}\}$, $k = k'$ and

$$s_i(\vec{c}, c_{\mathrm{w}}, c_{\mathrm{w}}) = \text{`}\langle v_i\,,\,v_i^+\rangle\text{'}, \quad \text{for } 1 \leq i \leq k. \tag{12}$$

Thus $s_i(\vec{a}, \text{`}\sqcup\text{'}, c_{\mathrm{w}})$ is either '$v_i$', '$v_i\sqcup$' (or '$v_i\sqcup\sqcup$'). Similarly, $s_i(\vec{b}, c_{\mathrm{w}}, \text{`}\sqcup\text{'})$ is either 'v_i^+', '$v_i^+\sqcup$' (or '$v_i^+\sqcup\sqcup$'). It follows from (9) that $q_0v_0 = v_1$, it follows from (11) that $v_k = q_{\mathrm{f}}$, and it follows from (10) that $v_i^+ = v_{i+1}$ for $1 \leq i < k$. Hence, M accepts v_0.

($\Leftarrow$) Assume that M accepts v_0. We will construct a substitution θ that solves $\{S^M_{\mathrm{sp}}(\text{`}q_0v_0\text{'}, F)\,,\,S^M_{\mathrm{mv}}(F, G)\}$. Consider a valid computation of M with input v_0:

$$((v_1, v_2), (v_2, v_3), \ldots, (v_k, v_{k+1})),$$

for some $k \geq 1$, where $v_1 = q_0v_0$, $v_i^+ = v_{i+1}$ for $1 \leq i \leq k$, and $v_k = q_{\mathrm{f}}$ (i.e., $v_{k+1} = \epsilon$). We define $F\theta$ as in (8) with $k' = k$, where the s_i's are defined as follows. First, let i, $1 \leq i \leq k$, be fixed and assume that

$$\langle v_i\,,\,v_i^+\rangle = c_{a_{j_1}b_{j_1}}c_{a_{j_2}b_{j_2}}\cdots c_{a_{j_n}b_{j_n}},$$

Let s_i be the term:

$$\lambda\vec{z}\,.\,f_{\mathrm{w}}(z_{j_1}, f_{\mathrm{w}}(z_{j_2}, \cdots f_{\mathrm{w}}(z_{j_n}, s_i')\cdots)),$$

where s_i' is defined as one of $\{x_1, x_2, c_{\mathrm{w}}\}$ according to the following rules. We write $|v|$ for the length of a string v. First, suppose that $1 < i < k$. There are in principle nine different cases (according to the lengths of the moves that surround the i'th move):

1. $|\langle v_{i-1}\,,\,v_{i-1}^+\rangle| < |\langle v_i\,,\,v_i^+\rangle| = |\langle v_{i+1}\,,\,v_{i+1}^+\rangle|$,
2. $|\langle v_{i-1}\,,\,v_{i-1}^+\rangle| = |\langle v_i\,,\,v_i^+\rangle| = |\langle v_{i+1}\,,\,v_{i+1}^+\rangle|$,
3. $|\langle v_{i-1}\,,\,v_{i-1}^+\rangle| > |\langle v_i\,,\,v_i^+\rangle| = |\langle v_{i+1}\,,\,v_{i+1}^+\rangle|$,
4. $|\langle v_{i-1}\,,\,v_{i-1}^+\rangle| < |\langle v_i\,,\,v_i^+\rangle| > |\langle v_{i+1}\,,\,v_{i+1}^+\rangle|$,
5. $|\langle v_{i-1}\,,\,v_{i-1}^+\rangle| = |\langle v_i\,,\,v_i^+\rangle| > |\langle v_{i+1}\,,\,v_{i+1}^+\rangle|$,
6. $|\langle v_{i-1}\,,\,v_{i-1}^+\rangle| > |\langle v_i\,,\,v_i^+\rangle| > |\langle v_{i+1}\,,\,v_{i+1}^+\rangle|$,
7. $|\langle v_{i-1}\,,\,v_{i-1}^+\rangle| < |\langle v_i\,,\,v_i^+\rangle| < |\langle v_{i+1}\,,\,v_{i+1}^+\rangle|$,
8. $|\langle v_{i-1}\,,\,v_{i-1}^+\rangle| = |\langle v_i\,,\,v_i^+\rangle| < |\langle v_{i+1}\,,\,v_{i+1}^+\rangle|$,

and the case $|\langle v_{i-1}\,,\, v_{i-1}^{+}\rangle| > |\langle v_i\,,\, v_i^{+}\rangle| < |\langle v_{i+1}\,,\, v_{i+1}^{+}\rangle|$, that is (due to the special assumptions about M) not possible.

Let s'_i be x_1 in cases 3 and 6. Let s'_i be x_2 in cases 7 and 8. In all the other cases (i.e., 1, 2, 4, and 5) let s'_i be c_{w}. Second, suppose that $i = 1$, we may assume that the first move has the same length as the second move, and let $s'_i = c_{\mathrm{w}}$. Third, suppose that $i = k$, we may assume that the last move is strictly shorter than the move before it, and let $s'_i = x_1$. Obviously,

$$s_i(\vec{c}, c_{\mathrm{w}}, c_{\mathrm{w}}) = \text{`}\langle v_i\,,\, v_{i+1}\rangle\text{'}, \quad (1 \le i \le k).$$

Let $F\theta$ be defined as in (8). Then the term $\theta(F(\vec{c}, c_{\mathrm{w}}, c_{\mathrm{w}}, c_{\mathrm{t}}))$ is a move-train of M. First, it follows from Lemma 12 that

$$\theta(F(\vec{c}, c_{\mathrm{w}}, c_{\mathrm{w}}, c_{\mathrm{t}})) \xrightarrow{*}_{R_{\mathrm{mv}}} c_{\mathrm{t}}.$$

Second, it follows from Lemma 2, that there exists a substitution θ that solves $S^M_{\mathrm{mv}}(F, G)$. Finally, one can easily check that the conditions (9), (10), and (11) hold, by looking at the different cases (corresponding to cases 1–8) above. For example, if case 3 holds for i (i.e., $s'_i = x_1$), then one of cases 4–6 holds for $i - 1$ (i.e., $s'_{i-1} = c_{\mathrm{w}}$ or $s'_{i-1} = x_1$), and thus, $s_i(\vec{a}, \text{`}\sqcup\text{'}, c_{\mathrm{w}}) = \text{`}v_{i\sqcup}\text{'}$ and $s_{i-1}(\vec{b}, c_{\mathrm{w}}, \text{`}\sqcup\text{'}) = \text{`}v_{i\sqcup}\text{'}$. It follows from Lemma 1 that θ also solves the equation $S^M_{\mathrm{sp}}(\text{`}q_0 v_0\text{'}, F)$. ■

Let us consider a fixed *universal Turing machine* M_{u} with input alphabet Σ_{u} and initial state q_0. Any pair (M, v), where M is a TM and v is an input string for M is encoded effectively as a string over Σ_{u}, denoted by $\langle M\,,\, v\rangle$. The details of such an encoding are not relevant here and can be found for example in [Hopcroft & Ullman 1979]. The universal TM accepts $\langle M\,,\, v\rangle$ if and only if M accepts v. The following corollary is an easy consequence of Theorem 14, if the constructions there are based on M_{u}.

Corollary 15 *There is system $S^{\mathrm{u}}(X, F, G)$ of two second-order equations:*

$$\{\, F(\vec{t_1}) \stackrel{?}{=} f(X, F(\vec{t_2})), \quad G(\vec{t_3}) \stackrel{?}{=} g(F(\vec{t_4}), G(\vec{t_5}))\,\},$$

where the $\vec{t_i}$'s are sequences of ground terms of depth ≤ 2, such that the problem of determining whether $S^{\mathrm{u}}(t, F, G)$ is solvable for a given ground term t, is undecidable.

¿From the structure of S^{u} and the results in Section 4 we can conclude the following. The integer n in Corollary 16 can easily be calculated from S^{u}.

Corollary 16 *There is an integer n such that second-order unification is undecidable for equation systems satisfying conditions (1–5):*

1. **(Schubert [1997])** *the arguments of all variables are ground terms,*
2. *the arguments of all variables have depth $< n$,*

3. *the arity of all variables is* $< n$,

4. *the number of occurrences of second-order variables is* ≤ 5, *and*

5. *one of the following two cases holds:*

   (a) *there is a single second-order variable, or*

   (b) *there are at most two second-order variables and no first-order variables.*

It is straightforward to show that Corollary 16 remains true even when the signature includes a single constant and a single function symbol of arity > 1, by using encoding techniques in [Farmer 1991].

It was observed by Voronkov that, by applying the reduction in [Degtyarev & Voronkov 1996] to a system of *simple* second-order equations, one obtains a system of reachability constraints with *ground left-hand sides*. (The converse does not hold, i.e., the reduction from a system of reachability constraints with ground left-hand sides, by using Theorem 5, does in general not yield a system of simple second-order equations.) Thus, by using Schubert's [1997] result, Degtyarev & Voronkov's [1996] reduction implies an elegant proof of Plaisted's [1995] result.

Corollary 17 (Plaisted [1995]) *SREU is undecidable with ground left-hand sides.*

6.4 Related work

A form of shifted pairing appears already in Hopcroft & Ullman's [1979] proof of the undecidability of the intersection emptiness problem of context free languages. It was more directly used by Plaisted [1995] in the context of SREU, and then in a refined form by Veanes [1996]. The Train Lemma [Gurevich & Veanes 1999] uses tree automata theoretic techniques and connections to term rewriting [e.g. Dauchet 1993] and has its roots in [Plaisted 1995, Veanes 1996]. These two techniques have been the main tools in proving undecidability results of restricted forms of SREU [e.g. Gurevich & Veanes 1999]. The proof of Theorem 14 is an application of these techniques in the context of SOU, by using the relations between rigid reachability and second-order unification developed in Section 3. In fact, some relations between rigid reachability and second-order unification were explicitly used already by Farmer [1991, Lemma 5.2]. A closer look at the multiplication equations in Goldfarb's [1981] proof shows that similar properties are also exploited there.

There is a close relationship between our work and the work of Schubert [1997]. Schubert showed the undecidability of SOU for simple equations. This result was an important tool in establishing the undecidability of certain type inference problems [Schubert 1997, Schubert 1998]. His undecidability proof is by reduction from Minsky machines and the construction is quite technical. Although our proof was developed independently, the resulting set of equations

in [Schubert 1997, Definition 17] includes two main equations whose shape and purpose have a close resemblance to those of the ones in Theorem 14. A careful analysis of the constructions in [Schubert 1997] shows that, they also rely upon links between simple equations of the form in Lemma 2 and ground term rewriting.

Further evidence of the fundamental role of such equations is that, solvability of a *single* equation of the form in Lemma 2 is already undecidable [Ganzinger et al. 1998] (also this result uses the shifted pairing technique).

7 Current status of second-order unification and some open problems

We give a (roughly) chronological list of the main results known about second-order unification, including the main new contributions of this paper. *Monadic* below means that all function symbols have arity ≤ 1. Recall also that in *simple* equations, arguments of second-order variables are required to be ground terms.

1. The undecidability of higher-order unification in general, in fact third-order unification, was proved (independently) by Huet [1973] and by Lucchesi [1972].

2. In [1981] Goldfarb showed that second-order unification is undecidable, by reduction from Hilbert's tenth problem.

3. In [1979] Zhezherun showed that monadic second-order unification is decidable. This result is also proved by Farmer [1988], by reduction to word equations (the decidability of the latter problem is an important result due to Makanin [1977]).

4. In [1991] Farmer showed that there is an integer n such that second-order unification is undecidable even if all second-order variables are unary, and there are at most n second-order variables, and first-order variables are not allowed to occur in equations.

5. Some (incomparable) cases of context unification are proved to be decidable by Comon [1993] and Schmidt-Schauß [1995]. Recent developments towards a more general decidability result are discussed by Schmidt-Schauß & Schulz [1998].

6. Some cases of linear second-order unification are proved to be decidable by Levy [1996], in particular when each variable occurs at most twice.

7. Schubert [1997] proves that second-order unification is undecidable for systems of simple equations.

8. A natural polynomial time reduction from second-order unification to SREU is given by Degtyarev & Voronkov [1996]. A converse direction, from SREU to second-order unification, is given in Section 4.

9. In Section 5 we show that second-order unification is undecidable already with a *single* unary second-order variable.

10. In Section 6 we give a simpler proof of Schubert's [1997] result and improve it in several ways. In particular, second-order unification is shown undecidable in the simple case already with a single second-order variable that is, moreover, only allowed to occur at most five times.

11. Ganzinger et al. [1998] prove that rigid reachability is undecidable, even for ground systems of rules. They use this result and an analogue of Lemma 2 to derive the undecidability of second-order unification with a single second-order variable that occurs at most twice.

The above list of results is not exhaustive. Some borders between decidable and undecidable cases of second-order unification are also pointed out by Prehofer [1994]. In this paper we have not discussed *matching* problems. It should be noted however, that the *second-order matching* problem was proved decidable by Huet & Lang [1978]. This result has been generalized in [Dowek 1991, Dowek 1994, Padovani 1996] and Comon & Jurski [1997] have shown some computational complexity results.

Moreover, due to the strong connection between second-order unification and SREU, several results concerning SREU, in particular its relation to intuitionistic logic and to several fundamental classical decision problems related to Herbrand's theorem, carry over to second-order unification. The most recent survey discussing such relations is given by [Voronkov 1998].

The decidability of monadic SREU is currently an open problem, only some special cases are known to be decidable [Degtyarev, Matiyasevich & Voronkov 1996, Gurevich & Voronkov 1997, Cortier, Ganzinger, Jacquemard & Veanes 1999]. In a recent paper Voronkov [1998] discusses connections of SREU with other problems and presents precise results with respect to signatures. The decidability of SREU restricted to *two* rigid equations, even in the *guarded* case and with *ground left-hand sides*, is another intriguing open problem. The corresponding fragment of simultaneous rigid reachability is undecidable [Levy & Veanes 1998]. In contrast, decidability of *guarded* (non-simultaneous) rigid reachability (i.e., when the rule set and one of the terms are ground) follows already from Brainerd's [1969] results.

Levy [1996] proved that *linear* second-order unification is decidable when no second-order variable occurs more than twice. In Section 4 we show that, under this same restriction, second-order unification is undecidable. This establishes a clear difference between these two apparently similar problems. The decidability of linear second-order unification, and context unification, are both open problems. In all undecidability proofs of the restricted cases of second-order unification that we have considered, a common key feature is the unboundedness of the number of occurrences of bound variables in closed terms in solutions. A recent result by Schmidt-Schauß [1998] shows that second-order unification becomes decidable, when an upper bound on the number of occurrences of each bound variable in a closed term in a solution is fixed.

Final remark. Since Goldfarb [1981] proved the undecidability of second-order unification, very few decidable and undecidable subclasses of second-order unification problems have been found. Here we have characterized decidability for classes defined in terms of number of second-order variables, number of occurrences per variable, arity of variables and groundness of their arguments (simplicity of equations).

Moreover, we have stated and proved a very close relationship between the simultaneous rigid E-unification problem and the second-order unification problem. This relationship allows us to translate some decidability/undecidability results and proof techniques from one class of problems to the other.

Acknowledgements

We would like to acknowledge M. Bonet, H. Ganzinger, A. Rubio, S. Vorobyov, M. Villaret and all the anonymous referees of this paper for their comments and support. We specially thank R. Nieuwenhuis and A. Voronkov for suggesting us the study of the possible relationship between second-order unification and the simultaneous rigid E-unification problem. We also thank A. Schubert for comments and suggestions that lead to considerable improvements of the final manuscript.

References

Baader, F. & Nipkow, T. (1998), *Term Rewriting and All That*, Cambridge University Press.

Brainerd, W. (1969), 'Tree generating regular systems', *Information and Control* **14**, 217–231.

Comon, H. (1993), Completion of rewrite systems with membership constraints, Technical report, CNRS and LRI, Université de Paris Sud. to appear in *J. Symbolic Computation*.

Comon, H. & Jurski, Y. (1997), Higher-order matching and tree automata, *in* M. Nielsen & W. Thomas, eds, 'Proc. Conf. on Computer Science Logic', Vol. 1414 of *LNCS*, Springer Verlag, Aarhus, pp. 157–176.

Cortier, V., Ganzinger, H., Jacquemard, F. & Veanes, M. (1999), Decidable fragments of simultaneous rigid reachability, submitted to *ICALP'99*.

Dauchet, M. (1993), Rewriting and tree automata, *in* H. Comon & J. P. Jouannaud, eds, 'Term Rewriting (French Spring School of Theoretical Computer Science)', Vol. 909 of *Lecture Notes in Computer Science*, Springer Verlag, Font Romeux, France, pp. 95–113.

Degtyarev, A. & Voronkov, A. (1995), Reduction of second-order unification to simultaneous rigid E-unification, Technical Report 109, Computer Science Department, Uppsala University.

Degtyarev, A. & Voronkov, A. (1996), 'The undecidability of simultaneous rigid E-unification', *Theoretical Computer Science* **166**(1-2), 291–300.

Degtyarev, A., Gurevich, Y. & Voronkov, A. (1996), Herbrand's theorem and equational reasoning: Problems and solutions, *in* 'Bulletin of the European Association for Theoretical Computer Science', Vol. 60. The "Logic in Computer Science" column.

Degtyarev, A., Matiyasevich, Y. & Voronkov, A. (1996), Simultaneous rigid E-unification and related algorithmic problems, *in* 'Eleventh Annual IEEE Symposium on Logic in Computer Science (LICS'96)', IEEE Computer Society Press, New Brunswick, NJ, pp. 494–502.

Dowek, G. (1991), A second-order pattern matching algorithm for the cube of typed λ–calculi, *in* A. Tarlecki, ed., 'Proceedings of Mathematical Foundations of Computer Science. (MFCS '91)', Vol. 520 of *Lecture Notes in Computer Science*, Springer Verlag, Kazimierz Dolny, Poland, pp. 151–160.

Dowek, G. (1994), 'Third order matching is decidable', *Annals of Pure and Applied Logic* **69**(2–3), 135–155.

Farmer, W. M. (1988), 'A unification algorithm for second-order monadic terms', *Annals of Pure and Applied Logic* **39**, 131–174.

Farmer, W. M. (1991), 'Simple second-order languages for which unification is undecidable', *Theoretical Computer Science* **87**, 173–214.

Gallier, J. H., Narendran, P., Plaisted, D. & Snyder, W. (1988), Rigid E-unification is NP-complete, *in* 'Proc. IEEE Conf. on Logic in Computer Science, LICS'88', pp. 338–346.

Gallier, J. H., Raatz, S. & Snyder, W. (1987), Theorem proving using rigid E-unification: Equational matings, *in* 'Proc. IEEE Conf. on Logic in Computer Science, LICS'87', pp. 338–346.

Ganzinger, H., Jacquemard, F. & Veanes, M. (1998), Rigid reachability, *in* J. Hsiang & A. Ohori, eds, 'Advances in Computing Science – ASIAN'98, 4th Asian Computing Science Conference, Manila, The Philippines, December 1998, Proceedings', Vol. 1538 of *Lecture Notes in Computer Science*, Springer Verlag, pp. 4–21.

Goldfarb, W. D. (1981), 'The undecidability of the second-order unification problem', *Theoretical Computer Science* **13**, 225–230.

Gurevich, Y. & Veanes, M. (1999), Logic with equality: Partisan corroboration, and shifted pairing, To appear in *Information and Computation*.

Gurevich, Y. & Voronkov, A. (1997), Monadic simultaneous rigid E-unification and related problems, *in* P. Degano, R. Corrieri & A. Marchetti-Spaccamella, eds, 'Automata, Languages and Programming, 24th International Colloquium, ICALP'97', Vol. 1256 of *Lecture Notes in Computer Science*, Springer Verlag, pp. 154–165.

Hindley, J. & Seldin, J. (1986), *Introduction to Combinatorics and λ-Calculus*, Cambridge University Press.

Hopcroft, J. E. & Ullman, J. D. (1979), *Introduction to Automata Theory, Languages and Computation*, Addison-Wesley Publishing Co.

Huet, G. (1973), 'The undecidability of unification in third-order logic', *Information and Control* **22**(3), 257–267.

Huet, G. & Lang, B. (1978), 'Proving and applying program transformations expressed with second-order patterns', *Acta Informatica* **11**, 21–55.

Levy, J. (1996), Linear second-order unification, *in* '7th Int. Conf. on Rewriting Techniques and Applications, RTA'96', Vol. 1103 of *Lecture Notes in Computer Science*, New Jersey, USA, pp. 332–346.

Levy, J. (1998), Decidable and undecidable second-order unification problems, *in* T. Nipkow, ed., 'Rewriting Techniques and Applications', Vol. 1379 of *Lecture Notes in Computer Science*, Springer Verlag, pp. 47–60.

Levy, J. & Veanes, M. (1998), On unification problems in restricted second-order languages, *in* 'Annual Conference of the European Association for Computer Science Logic, CSL'98', Brno, Czech Republic.

Lucchesi, C. L. (1972), The undecidability of the unification problem for third-order languages, Technical Report CSRR 2059, Dept. of Applied Analysis and Computer Science, Univ. of Waterloo.

Makanin, G. S. (1977), 'The problem of solvability of equations in a free semigroup', *Math. USSR Sbornik* **32**(2), 129–198.

Matiyasevich, Y. (1970), 'The diophantiness of recursively enumerable sets (in Russian)', *Soviet Mathematical Doklady* pp. 279–282.

Niehren, J., Pinkal, M. & Ruhrberg, P. (1997), On equality up-to constraints over finite trees, context unification, and one-step rewriting, *in* W. McCune, ed., 'Proceedings of the 14th International Conference on Automated Deduction', Vol. 1249 of *Lecture Notes in Artificial Intelligence*, Springer, Townsville, North Queensland, Australia, pp. 34–48.

Padovani, V. (1996), Filtrage d'ordre superieur, PhD thesis, Université Paris VII.

Pietrzykowski, T. (1973), 'A complete mechanization of second-order logic', *J. of the ACM* **20**(2), 333–364.

Plaisted, D. A. (1995), Special cases and substitutes for rigid E-unification, Technical Report MPI-I-95-2-010, Max-Planck-Institut für Informatik.

Prehofer, C. (1994), Decidable higher-order unification problems, *in* A. Bundy, ed., '12th International Conference on Automated Deduction', Vol. 814 of *Lecture Notes in Artificial Intelligence*, Springer Verlag, Nancy, France, pp. 635–649.

Schmidt-Schauß, M. (1995), Unification of stratified second-order terms, Technical Report 12/94, Johan Wolfgang-Goethe-Universität, Frankfurt, Germany.

Schmidt-Schauß, M. (1998), Decidability of bounded second order unification, Draft, Fachbereich Informatik, Johan Wolfgang-Goethe-Universität, Frankfurt, Germany.

Schmidt-Schauß, M. & Schulz, K. (1998), Decidability of context unification with two variables, *in* 'UNIF 98, 12th International Workshop on Unification, Rome, 29/06–01/07 1998', number SI-98/8 *in* 'Research Report', University of Rome, pp. 71–74. A more detailed version appears in *CADE'99*.

Schubert, A. (1997), Second-order unification and type inference for church-style polymorphism, Technical Report TR 97-02(239), Institute of Informatics, Warsaw University.

Schubert, A. (1998), Second-order unification and type inference for Church-style polymorphism, *in* 'Conference Record of POPL'98: The 25TH ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages', ACM Press, San Diego, California, pp. 279–288.

Schulz, K. U. (1991), Makanin's algorithm, two improvements and a generalization, Technical Report CIS-Bericht-91-39, Centrum für Informations und Sprachverarbeitung, Universität München.

Veanes, M. (1996), Uniform representation of recursively enumerable sets with simultaneous rigid E-unification, UPMAIL Technical Report 126, Uppsala University, Computing Science Department.

Veanes, M. (1998), The relation between second-order unification and simultaneous rigid E-unification, *in* 'Proc. Thirteenth Annual IEEE Symposium on Logic in Computer Science, June 21–24, 1998, Indianapolis, Indiana (LICS'98)', IEEE Computer Society Press, pp. 264–275.

Voronkov, A. (1998), Simultaneous rigid E-unification and other decision problems related to the Herbrand theorem, Technical Report 152, Uppsala University, Computing Science Department. Submitted to *Theoretical Computer Science*.

Zhezherun, A. P. (1979), 'Decidability of the unification problem for second-order languages with unary functional symbols', *Kibernetika (Kiev)* **5**, 120–125. Translated as *Cybernetics* 15(5): 735–741, 1980.

Legends

Figure 1: See the equation in Example 1. The top term is $\theta(F)$. The left (right) term is obtained by applying θ to the left (right) side of the equation. The encoded rewrite steps are indicated in grey.

Figure 2: See the equation in Example 2. The top term is $\theta(F)$. The left (right) term is obtained by applying θ to the left (right) side of the equation. The arguments of F are shown in gray.

Figure 3: Shifted pairing.

Figure 4: A closed term that encodes the valid computation in Example 6.

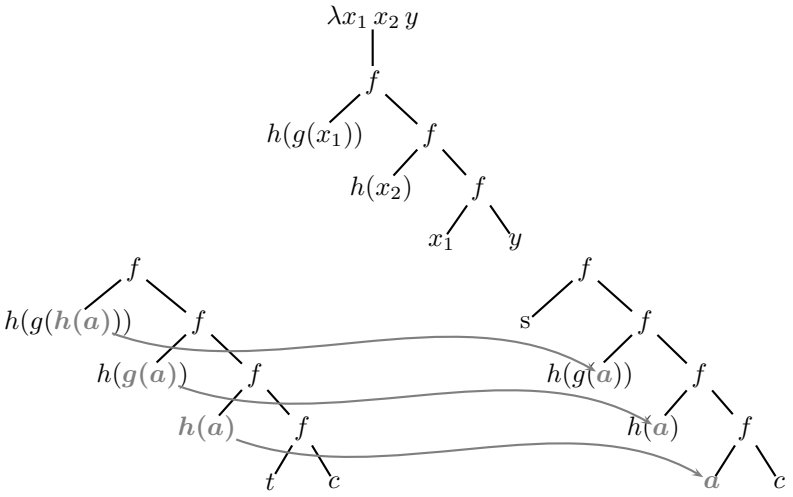
$\lambda x_1\, x_2\, y$
f
$h(g(x_1))$
f
$h(x_2)$
f
x_1
y
f
$h(g(h(a)))$
f
$h(g(a))$
f
$h(a)$
f
t
c
f
s
f
$h(g(a))$
f
$h(a)$
f
d
c

Figure 1:

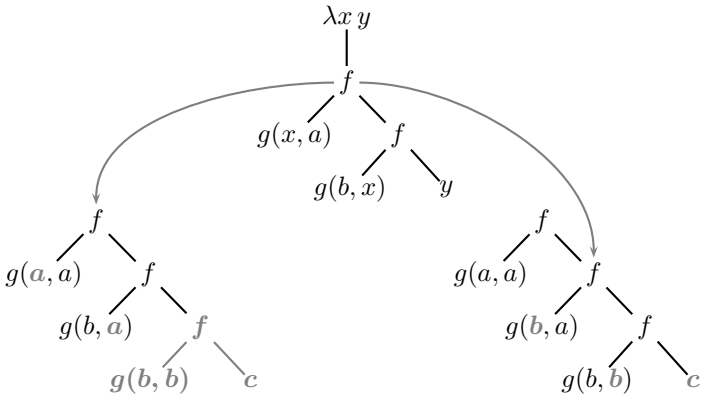

Figure 2:

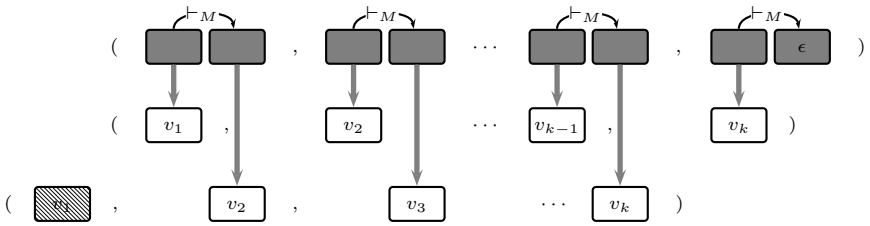

Figure 3:

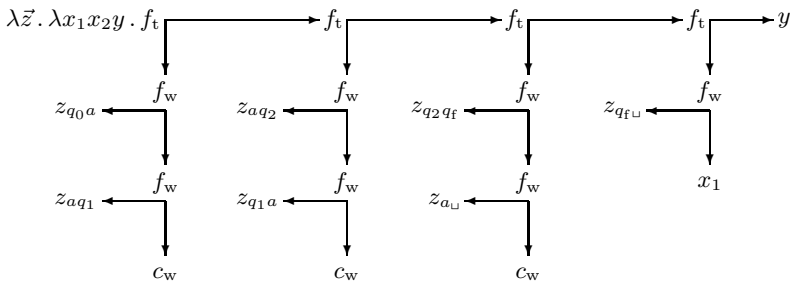
$\lambda\vec{z}\,.\,\lambda x_1 x_2 y\,.\,f_{\mathrm{t}}$
f_{t}
f_{t}
f_{t}
y
f_{w}
f_{w}
f_{w}
f_{w}
$z_{q_0 a}$
$z_{a q_2}$
$z_{q_2 q_{\mathrm{f}}}$
$z_{q_{\mathrm{f}} \sqcup}$
f_{w}
f_{w}
f_{w}
x_1
$z_{a q_1}$
$z_{q_1 a}$
$z_{a_\sqcup}$
c_{w}
c_{w}
c_{w}

Figure 4: