I N F O R M A T I K

The Relation Between Second-Order Unification and Simultaneous Rigid E-Unification

Margus Veanes

MPI–I–98–2-005 February 1998

FORSCHUNGSBERICHT RESEARCH REPORT

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

Im Stadtwald 66123 Saarbrücken Germany

Author's Address

Max-Planck-Institut für Informatik
Im Stadtwald
66123 Saarbrücken
Germany
`veanes@mpi-sb.mpg.de`
`http://www.mpi-sb.mpg.de/~veanes`

Acknowledgements

I thank Andrei Voronkov for many valuable discussions and for suggesting me to study the connections between second-order unification and simultaneous rigid E-unification, that ultimately led to this report. I thank Harald Ganzinger and Sergei Vorobyov for many valuable discussions and comments on earlier versions of this report.

Abstract

Simultaneous rigid E-unification, or SREU for short, is a fundamental problem that arises in global methods of automated theorem proving in classical logic with equality. In order to do proof search in intuitionistic logic with equality one has to handle SREU. Furthermore, restricted forms of SREU are strongly related to word equations and finite tree automata. Higher-order unification has applications in proof theory, computational linguistics, program transformation, and also in theorem proving. It was recently shown that second-order unification has a very natural reduction to simultaneous rigid E-unification, which constituted probably the most transparent undecidability proof of SREU. Here we show that there is also a natural encoding of SREU in second-order unification. It follows that the problems are logspace equivalent. We exploit this connection and use finite tree automata techniques to prove that second-order unification is undecidable in more restricted cases than known before. We present a more elementary undecidability proof of second-order unification than the previously known proofs exposing that already a very small fragment of second-order unification has the universal computational power.

Keywords

Second-order unification, simultaneous rigid E-unification, finite tree automata.

1 Introduction

Simultaneous Rigid E-Unification, or SREU for short, is originally introduced in 1987 by Gallier, Raatz, and Snyder [15] as a fundamental problem that arises in global or rigid-variable methods of automated theorem proving in classical logic with equality. For example, in free variable tableau methods, SREU corresponds to the problem of deciding if some substitution closes a given set of branches. In intuitionistic logic with equality, proof search leads to SREU. It is shown in Voronkov [33] that SREU is actually polynomial-time equivalent to *skeleton instantiation* (the problem of deciding if there is a proof with a given proof skeleton) in certain proof systems in intuitionistic logic with equality. There is a list of several fundamental decision problems in both classical and intuitionistic logic with equality that are very closely related to SREU [5].

The undecidability of SREU is shown in 1995 by Degtyarev and Voronkov [8]. Probably the most transparent undecidability proof of SREU [7, 10] is by reduction from *second-order unification*, proved undecidable in Goldfarb [17]. This reduction shows that any semidecision algorithm for SREU can be adapted to second-order unification. It was observed by the author and Voronkov that this connection should be studied further, in particular that the techniques used to prove some undecidability results of SREU [26, 29], could probably be adapted in the context of second-order unification. By using a recent construction from Levy [22], we present two lemmas in Section 3 that enable us to transfer techniques from the context of SREU to the context of SOU. The first application of these lemmas is the following result in Section 4, by using a restricted form of SREU [18].

SREU and second-order unification are logspace equivalent.

The general content of this result is that second-order unification plays the same fundamental role as SREU in logic with equality. For example, it implies that it is necessary to handle second-order unification in order to do proof search in intuitionistic logic with equality. From the viewpoint of automated theorem proving or proof search this connection is important because currently there are no known reasonable semidecision algorithms for SREU, except for ones based of straightforward enumeration [25]. The reduction to higher-order unification may provide new insights into how to deal with this.

By using a recent undecidability of a very restricted case of SREU [18], we obtain the following statement, that strengthens the result in Levy [22] with the last two conditions:

Second-order unification is undecidable with the following restrictions on equation systems $\mathcal{S}$:

1. *each second-order variable occurs at most twice in $\mathcal{S}$,*
2. *there are at most 3 second-order variables in $\mathcal{S}$,*
3. *there are at most 2 first-order variables in $\mathcal{S}$.*

There are many known relationships between SREU and *intuitionistic logic* [9, 31, 32], *finite tree automata* [16] and SREU [3, 4, 30], *word equations* [23] and *monadic* SREU [6, 19], word equations and *monadic* second-order unification [12], and there are connections between other restricted forms of second-order unification [21] and *context-unification* [27]. In Section 5 we exploit some of these connections and the lemmas in Section 3 to give a new elementary undecidability proof of second-order unification. In particular, we apply *finite tree automata* techniques in combination with encoding techniques involving valid Turing machine computations underlying the proofs in [18, 31]. To be precise, we construct (effectively) a *universal second-order equation* $S^{\mathrm{u}}(x, F, G)$ of the form

$$g(F(\vec{t}_1), G(\vec{t}_2)) \approx g(f(x, F(\vec{t}_3)), g(F(\vec{t}_4), G(\vec{t}_5)))$$

where all the $\vec{t}_i$'s are sequences of ground terms of depth ≤ 1 over a signature Σ_{u}, and F and G are second-order variables with matching arities, such that the following decision problem is *undecidable*:

Given a ground term s, is $S^{\mathrm{u}}(s, F, G)$ solvable?

The construction of the universal second-order equation is relatively short and involves two steps that are intuitively clear. We can conclude the following.

Second-order unification is undecidable with the following restrictions on equation systems $\mathcal{S}$:

1. *there are no first-order variables in $\mathcal{S}$,*
2. *at most two variables in $\mathcal{S}$,*
3. *at most five occurrences of variables in $\mathcal{S}$,*
4. *all occurrences of variables in $\mathcal{S}$ have ground arguments, and*
5. *all arguments of variables have depth $\leq k$ for some fixed $k \geq 1$.*

By applying certain encoding techniques in Farmer [13], we prove in Section 6 that a signature consisting of one constant and one binary function symbol is already enough to obtain undecidability, without violating the above restrictions. Hence, our result implies the following complement to Farmer's undecidability theorem:

> *There is an integer n such that for all nonmonadic second-order languages with at least two second-order variables with arities $\geq n$ the unification problem is undecidable already if there are at most five variable occurrences all having ground arguments of depth $\leq n$.*

The value of n is any integer greater than the maximum of the arities of F and G (in the universal second-order equation) and the value of k. Moreover, it seems that the use of two second-order variables is necessary to obtain undecidability, e.g., the problem is decidable if there is just one second-order variable and it occurs at most twice [22].

Our result also confirms the statement in Schubert [28] that the undecidability of second-order unification holds for systems of equations where all variables have ground arguments, and improves it with the conditions (1–3) and (5). Allowing variables to occur in arguments of second-order variables is essential in Goldfarb's proof [17]. We note that the proof by Schubert involves a very complicated reduction from Minsky machines and appears to have some gaps. The undecidability of second-order unification for equations where all variables have ground arguments is used in Schubert [28] to derive the undecidability of a certain type inference problem.[1]

2 Preliminaries

We assume that the reader is familiar with the notions of (first-order) terms, equations, substitutions, and standard notions related to first-order logic. We define the corresponding second-order notions without using an explicit variable binding operator like λ, following Farmer [13] or Goldfarb [17].

A *signature* Σ is a collection of *function symbols* with fixed arities ≥ 0 and, unless otherwise stated, Σ is assumed to contain at least one *constant* or function symbol with arity 0. We use $(a, b, c, d, a_1, \ldots)$ for constants and $(f, g, f_1, \ldots)$ for function symbols in general. A designated constant in Σ is denoted by c_Σ.

[1]Schubert shows the undecidability of the following decision problem for the Church-style system $\mathcal{F}$ [1]: Given a term t, does there exist a base Γ and a type τ such that $\Gamma \vdash t : \tau$ is correct in the system $\mathcal{F}$?

A *term language* or simply *language* is a triple $L = (\Sigma_L, \mathcal{X}_L, \mathcal{F}_L)$ of pairwise disjoint sets of symbols, where

- Σ_L is a signature,
- $\mathcal{X}_L$ $(x, y, x_1, y_1, \ldots)$ is a collection of first-order variables, and
- $\mathcal{F}_L$ $(F, G, F_1, F', \ldots)$ is a collection of symbols with fixed arities ≥ 1, called *second-order variables*.

Let L be a language. L is *first-order*, if $\mathcal{F}_L$ is empty; L is *second-order*, otherwise. If L is first-order then we write it as the pair $(\Sigma_L, \mathcal{X}_L)$. L is *monadic* if all function symbols in Σ_L have arity ≤ 1. We use $(\nu, \nu_1, \ldots)$ to denote arbitrary variables in L, i.e., symbols in $\mathcal{X}_L \cup \mathcal{F}_L$. A language L_1 is an *expansion* of a language L, in symbols $L_1 \supseteq L$ or $L \subseteq L_1$, if $\Sigma_L \subseteq \Sigma_{L_1}$, $\mathcal{X}_L \subseteq \mathcal{X}_{L_1}$, and $\mathcal{F}_L \subseteq \mathcal{F}_{L_1}$.

The set of all terms in a language L, or *L-terms*, is denoted by $\mathcal{T}_L$ and is defined as the set of all terms in the first-order language $(\Sigma_L \cup \mathcal{F}_L, \mathcal{X}_L)$. We use $(s, t, l, r, s_1, \ldots)$ for terms. We usually omit mentioning L when it is clear from the context. The *depth* of a term is defined as usual, by letting the depth of a constant or first-order variable be 0 and the depth of a compound term be 1 plus the maximum of the depths of its immediate subterms. A *ground* term is one that contains no variables. The set of all ground terms in a language L is denoted by $\mathcal{T}_{\Sigma_L}$. Given a term $F(\vec{t})$, where F is a second-order variable with arity m and $\vec{t}$ is a sequence of m terms, the elements of $\vec{t}$ are called the *arguments* of F. A (second-order) term is called *simple* if there are no nested occurrences of variables in it, i.e., all occurrences of second-order variables have ground arguments.

An *equation in L* is an unordered pair of L-terms, denoted by $s \approx t$. Equations are denoted by $(e, e_1, \ldots)$. A *rule in L* is an ordered pair of L-terms, denoted by $s \rightarrow t$.[2] An equation or a rule is *ground* (*simple*) if the terms in it are ground (simple). The *depth* of an equation $s \approx t$ is the maximum of the depths of s and t. A *system* of rules or equations is a finite set of rules or equations. Let R be a system of ground rules, and s and t two ground terms. Then s *rewrites* (in R) to t, denoted by $s \longrightarrow_R t$, if t is obtained from s by replacing an occurrence of a term l in s by a term r for some rule $l \rightarrow r$ in R. The term s *reduces* (in R) to t, denoted by $s \xrightarrow{*}_R t$, if either $s = t$ or s rewrites to a term that reduces to t. We assume that the the reader is familiar with the basic concepts in ground rewriting [11].

[2]By rules we understand thus *directed equations*. Only ground instantiations of rules are considered as *rewrite rules*.

2.1 Second-Order Unification

Given a language L, we need expressions representing functions that produce instances of terms in L. For that purpose we introduce an expansion L^* of L. We follow Goldfarb [17] and Farmer [13]. Let $\{z_i\}_{i\geq 1}$ be an infinite collection of new symbols not in L. The language L^* differs from L by having $\{z_i\}_{i\geq 1}$ as additional first-order variables, called *bound variables*. The *rank* of a term t in L^*, is either 0 if t contains no bound variables (i.e., $t \in \mathcal{T}_L$), or the largest n such that z_n occurs in t. Given terms t and $t_1, t_2, \ldots, t_n$ in L^*, we write $t[t_1, t_2, \ldots, t_n]$ for the term that results from t by simultaneously replacing z_i in it by t_i for $1 \leq i \leq n$. An L^*-term is called *closed* if it contains no variables other than bound variables. Note that closed L^*-terms of rank 0 are ground L-terms.

A *substitution in L* is a function θ with finite domain $\mathrm{dom}(\theta) \subseteq \mathcal{X}_L \cup \mathcal{F}_L$ that maps first-order variables to L-terms, and n-ary second-order variables to L^*-terms of rank $\leq n$. A substitution θ with domain $\{\, \nu_i \mid 1 \leq i \leq n \,\}$ such that $\theta(\nu_i) = t_i$ for $1 \leq i \leq n$, is also denoted by $\{\, \nu_i \mapsto t_i \mid 1 \leq i \leq n \,\}$. The result of applying a substitution θ to an L-term s, denoted by $s\theta$, is defined by induction on s:

1. If $s = x$ and $x \in \mathrm{dom}(\theta)$ then $s\theta = \theta(x)$.
2. If $s = x$ and $x \notin \mathrm{dom}(\theta)$ then $s\theta = x$.
3. If $s = F(t_1, \ldots, t_n)$ and $F \in \mathrm{dom}(\theta)$ then $s\theta = \theta(F)[t_1\theta, \ldots, t_n\theta]$.
4. If $s = F(t_1, \ldots, t_n)$ and $F \notin \mathrm{dom}(\theta)$ then $s\theta = F(t_1\theta, \ldots, t_n\theta)$.
5. If $s = f(t_1, \ldots, t_n)$ then $s\theta = f(t_1\theta, \ldots, t_n\theta)$.

We write also $F\theta$ for $\theta(F)$, where F is a second-order variable. A substitution is called *closed*, if its range is a set of closed terms. Given a term t, a substitution θ is said to be *grounding for t* if $t\theta$ is ground, similarly for other L-expressions. Given a sequence $\vec{t} = t_1, \ldots, t_n$ of terms, we write $\vec{t}\theta$ for $t_1\theta, \ldots, t_n\theta$.

Let E be a system of equations in L. A *unifier* of E is a substitution θ (in L) such that $s\theta = t\theta$ for all equations $s \approx t$ in E. E is *unifiable* if there exists a unifier of E. Note that if E is unifiable then it has a closed unifier that is grounding for E, since $\mathcal{T}_{\Sigma_L}$ is nonempty. The *unification problem for L* is the problem of deciding whether a given equation system in L is unifiable. In general, the *second-order unification* problem or *SOU* is the unification problem for arbitrary second-order languages. *Monadic* SOU is SOU for monadic second-order languages.

2.2 Simultaneous Rigid E-Unification

In the following let L be a first-order language. Given a system of equations E in L and an equation e in L, the expression $E \vdash_\forall e$ is called a *rigid equation* in L, where E is called the *left-hand side* of $E \vdash_\forall e$. A rigid equation $E \vdash_\forall e$ is *solvable* if there exists a substitution θ that is grounding for E and e such that $e\theta$ is a logical consequence of $E\theta$, such a substitution is said to *solve* the rigid equation. *Rigid E-unification* is the problem of deciding if a given rigid equation is solvable.

A *system* or finite set of rigid equations is *solvable*, if there exists a substitution that solves each rigid equation in that system. *Simultaneous rigid E-unification for L* or *SREU for L* is the problem of deciding if a given system of rigid equations in L is solvable. In general, by SREU we mean SREU for arbitrary first-order languages. *Monadic* SREU is SREU for monadic first-order languages.

3 Relations between rewriting and second-order unification

In this section we present two lemmas that show a close relationship between certain forms of rewriting and second-order unification. These lemmas are used as basic tools in the following sections. The main statement is Lemma 1, that is inspired by the proof of the main lemma in Levy [22] and is used to derive a strengthened version of the latter (Lemma 2) with analogous proof. The basic techniques that are involved in these constructions appear already in Farmer [13] and Goldfarb [17].

We frequently need to refer to certain conditions on a sequence of parameters. In order to avoid lengthy repetitions of such conditions we use the following definition. We say that a parameter sequence

$$(L, c, f, F, m, L_1, s, t, \vec{l}, \vec{r})$$

is *appropriate* if the following conditions hold:

- L is a language;
- c is a constant, f is a binary function symbol, and $f, c \notin \Sigma_L$;
- F is a second-order variable with arity $m+1$, $m \geq 0$, $F \notin \mathcal{F}_L$;
- $L_1 \supseteq (\Sigma_L \cup \{f, c\}, \mathcal{X}_L, \mathcal{F}_L \cup \{F\})$;

- $s, t \in \mathcal{T}_L$, $\vec{l} = l_1, \ldots, l_m \in \mathcal{T}_L$, and $\vec{r} = r_1, \ldots, r_m \in \mathcal{T}_L$.

Given appropriate $(L, c, f, F, m, L_1, s, t, \vec{l}, \vec{r})$, we use the following construction from Levy [22]. Let $SOE(c, f, F, s, t, \vec{l}, \vec{r})$ denote the following second-order equation in L_1:

$$SOE(c, f, F, s, t, \vec{l}, \vec{r}) \quad = \quad F(\vec{l}, f(s, c)) \approx f(t, F(\vec{r}, c)).$$

Note that the sequences $\vec{l}$ and $\vec{r}$ are empty when $m = 0$.

Lemma 1 *Let $(L, c, f, F, m, L_1, s, t, \vec{l}, \vec{r})$ be appropriate. The following statements are equivalent for all θ in L_1 such that $\vec{l}\theta, s\theta \in \mathcal{T}_{\Sigma_L}$.*

(i) θ solves $SOE(c, f, F, s, t, \vec{l}, \vec{r})$, i.e., $F\theta[\vec{l}\theta, f(s\theta, c)] = f(t\theta, F\theta[\vec{r}\theta, c])$.

(ii) Either $t\theta = s\theta$ and $F\theta = z_{m+1}$, or there exists $k \geq 1$ and closed L^*-terms s_i of rank $\leq m$ for $1 \leq i \leq k$, such that

$$F\theta = f(s_1, f(s_2, \ldots, f(s_k, z_{m+1}) \cdots)) \tag{1}$$

and

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

Proof.

(i)⇐(ii) Straightforward.

(i)⇒(ii) Let θ satisfying (i) be given. Say that an L_1^*-term t' is a *list* if either $t' = z_{m+1}$, or $t' = f(t_1, t_2)$ for some t_1 and t_2, where t_2 is a list. We use the following statement to show that $F\theta$ is a list:

(*) *For all L_1^*-terms t_1 and t_2, if*

$$t_2[\vec{l}\theta, f(s\theta, c)] = f(t_1, t_2[\vec{r}\theta, c]) \tag{5}$$

then t_2 is a list.

Proof. By induction on the number of symbols in t_2.

Assume that t_2 is either a constant, a variable in $\mathcal{X}_{L_1}$, or a bound variable. If (5) holds then the head symbol of the left-hand side of (5) must be f. The only possibility is $t_2 = z_{m+1}$ because $\vec{l}\theta \in \mathcal{T}_{\Sigma_L}$ and $f \notin \Sigma_L$.

Assume now that t_2 is a compound term and (5) holds. Then the head symbol of t_2 must be f. So $t_2 = f(t_{21}, t_{22})$ for some L_1^* terms t_{21} and t_{22}. Hence, by (5),

$$f(t_{21}, t_{22})[\vec{l}\theta, f(s\theta, c)] = f(t_1, f(t_{21}, t_{22})[\vec{r}\theta, c]),$$

and thus

$$t_{22}[\vec{l}\theta, f(s\theta, c)] = f(t_{21}[\vec{r}\theta, c], t_{22}[\vec{r}\theta, c]).$$

By the induction hypothesis t_{22} is a list and hence so is t_2. ⊠

We know that $F\theta$ is an L_1^*-term of rank $m + 1$. It follows from (*) and (i) that $F\theta$ is a list. If $F\theta = z_{m+1}$ then (i) implies that $s\theta = t\theta$, and thus (ii) holds. Assume that $F\theta \neq z_{m+1}$, i.e., there exists $k \geq 1$ and L_1^*-terms s_i (of rank $\leq m + 1$) for $1 \leq i \leq k$, such that

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

We show that each s_i is a closed L^*-term of rank $\leq m$, such that (2–4) hold. It follows from (i) that

$$\begin{array}{llllll} f(s_1[\vec{l}\theta, s'], & \ldots & f(s_{i+1}[\vec{l}\theta, s'], & \ldots & f(s\theta, & c)\cdots)\cdots) = \\ f(t\theta, & \ldots & f(s_i[\vec{r}\theta, c], & \ldots & f(s_k[\vec{r}\theta, c], & c)\cdots)\cdots), \end{array} \tag{6}$$

where $s' = f(s\theta, c)$. Hence, $s_1[\vec{l}\theta, s'] = t\theta$, $s_{i+1}[\vec{l}\theta, s'] = s_i[\vec{r}\theta, c]$ for $1 \leq i < k$, and $s\theta = s_k[\vec{r}\theta, c]$. So, s_k is a closed L^*-term of rank $\leq m$ since $s\theta$ is a ground L-term and c is not in L (recall that $\vec{r}$ is a sequence of m terms), and thus (4) holds. We prove by induction on $k - i$ that each s_i is a closed L^*-term of rank $\leq m$. The base case is $i = k$. Assume the statement is true for $i + 1 \leq k$, we prove it for i. Then $s_{i+1}[\vec{l}\theta, s'] = s_{i+1}[\vec{l}\theta]$ is a ground L-term since $\vec{l}\theta$ are ground L-terms, and by above, so is $s_i[\vec{r}\theta, c]$. Hence, s_i is a closed L^*-term of rank $\leq m$, since c is not in L. The conditions (2) and (3) follow. ⊠

Lemma 2 *Let $(L, c, f, F, m, L_1, s, t, \vec{l}, \vec{r})$ be appropriate. The following statements are equivalent for all θ in L_1 such that $F \notin \mathrm{dom}(\theta)$ and $s\theta, \vec{l}\theta, \vec{r}\theta \in \mathcal{T}_{\Sigma_L}$.*

(i) $\theta \cup \{F \mapsto t'\}$ solves $SOE(c, f, F, s, t, \vec{l}, \vec{r})$ for some t'.

(ii) ($t\theta \in \mathcal{T}_{\Sigma_L}$ and) $t\theta \xrightarrow{*}_{\{l_i\theta \to r_i\theta \mid 1 \leq i \leq m\}} s\theta$.

Proof.

((i)⇒(ii)) Let $\theta' = \theta \cup \{F \mapsto t'\}$ satisfying (i) be given. There are two cases by Lemma 1((i)⇒(ii)). First case is $t' = z_{m+1}$ and $t\theta' = s\theta'$, and thus $t\theta = s\theta$ (since $s, t \in \mathcal{T}_L$ and $F \notin \mathcal{F}_L$), and hence (ii) holds trivially. The second case is that there exists $k \geq 1$ and closed L^*-terms s_i of rank $\leq m$ for $1 \leq i \leq k$, such that

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

and $t\theta = s_1[\vec{l}\theta]$, $s_i[\vec{r}\theta] = s_{i+1}[\vec{l}\theta]$, for $1 \leq i < k$, $s_k[\vec{r}\theta] = s\theta$, where we have correctly inserted θ for θ' since $s, t, \vec{l}, \vec{r} \in \mathcal{T}_L$ and $F \notin \mathcal{F}_L$. Let R be the following system of ground rules in L:

$$R = \{\, l_i\theta \to r_i\theta \mid 1 \leq i \leq m \,\}.$$

Clearly, $s_i[\vec{l}\theta] \xrightarrow{*}_R s_i[\vec{r}\theta]$ for $1 \leq i \leq k$, and thus $s_i[\vec{l}\theta] \xrightarrow{*}_R s_{i+1}[\vec{l}\theta]$, for $1 \leq i < k$, by above. It follows that

$$t\theta = s_1[\vec{l}\theta] \xrightarrow{*}_R s_k[\vec{l}\theta] \xrightarrow{*}_R s_k[\vec{r}\theta] = s\theta,$$

as needed.

((ii)⇒(i)) Let θ satisfying (ii) be given and $R = \{\, l_i\theta \to r_i\theta \mid 1 \leq i \leq m \,\}$. We have a reduction:

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

If $k = 0$ then let $t' = z_{m+1}$ and (i) follows from Lemma 1((ii)⇒(i)). Assume that $k \geq 1$ and consider a fixed i, $1 \leq i \leq k$. The rewrite step $t_{i-1} \longrightarrow_R t_i$ uses a rule $l_j\theta \to r_j\theta$ for some j, $1 \leq j \leq m$, 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 by z_j. Thus $s_i[\vec{l}\theta] = t_{i-1}$ and $s_i[\vec{r}\theta] = t_i$. Given such s_i for $1 \leq i \leq k$, obviously (2–4) are true. Let t' be the term above. Statement (i) follows from Lemma 1((ii)⇒(i)). ⊠

4 Reduction of SREU to SOU

In this section we show that there is a logspace reduction of SREU to SOU. The converse reduction, from SOU to SREU, is given in Degtyarev and Voronkov [7, 10] and is also logspace. We use a restricted form of SREU that is logspace equivalent with SREU and allows us to apply Lemma 2 in a direct manner.

Let L be a first-order language and $\mathcal{R}$ a system of rigid equations in L and let L_1 be an arbitrary expansion of L. The following property for systems of rigid equations guarantees that any substitution θ in L_1 that solves $\mathcal{R}$, maps variables occurring in $\mathcal{R}$ to ground terms in L. Let x be a variable that occurs in $\mathcal{R}$. A *guard for x in $\mathcal{R}$*, if one exists, is any rigid equation $E \vdash_{\forall} s \approx t$ in $\mathcal{R}$ such that

- E is a set of ground equations,
- s is a ground term, and
- x occurs in t.

The system $\mathcal{R}$ is called *guarded* if there is a guard in $\mathcal{R}$ for each variable that occurs in $\mathcal{R}$. *Guarded SREU* is SREU restricted to guarded systems of rigid equations. The notion of guardedness is introduced in Gurevich and Veanes [18].

Lemma 3 *SREU is logspace equivalent to guarded SREU.*

Proof. Let $\mathcal{R}$ be a system of rigid equations and Σ the set of function symbols in $\mathcal{R}$ expanded with an additional constant if there is no constant in $\mathcal{R}$. Let $\mathcal{X}$ be the set of variables in $\mathcal{R}$ and let $L = (\Sigma, \mathcal{X})$. For each variable x in $\mathcal{X}$, for which there is no guard in $\mathcal{R}$, construct the following rigid equation:[3]

$$Gr(\Sigma, x) \quad = \quad \{\, f(c_\Sigma, \ldots, c_\Sigma) \approx c_\Sigma \mid f \in \Sigma \setminus \{c_\Sigma\} \,\} \vdash_{\forall} c_\Sigma \approx x.$$

Let $\mathcal{R}'$ be the extension of $\mathcal{R}$ with such rigid equations. Obviously is $\mathcal{R}'$ guarded. It is straightforward to prove that for all substitutions θ, θ solves $Gr(\Sigma, x)$ if and only if $x\theta \in \mathcal{T}_\Sigma$ [10, Lemma 3]. Hence, for all substitutions θ in L, θ solves $\mathcal{R}$ if and only if θ solves $\mathcal{R}'$. ⊠

The following result is implied by the reduction in Degtyarev and Voronkov [10].[4]

Theorem 1 (Degtyarev-Voronkov [10]) *There is a logspace reduction of SOU to SREU.*

[3]Note that $f(c_\Sigma, \ldots, c_\Sigma)$ stands for f when f has arity 0, it stands for $f(c_\Sigma)$ when f has arity 1, and so on.

[4]Personal communication with Voronkov.

We use the following definitions. Let L be a first-order language and $\mathcal{R}$ a system of rigid equations in L. For an equation system E in L we write R_E for the following set of rules in L:

$$R_E = \{ s \to t, t \to s \mid s \approx t \in E \}.$$

Let $\mathcal{F}_{\mathcal{R}}$ denote the following set of distinct second-order variables:

$$\mathcal{F}_{\mathcal{R}} = \{ F_{E \vdash_{\forall} e} \text{ of arity } |R_E| + 1 \mid E \vdash_{\forall} e \in \mathcal{R} \}.$$

Let $f_{\mathcal{R}}$ be a binary function symbol and $c_{\mathcal{R}}$ a constant that are not in L. We write $L_{\mathcal{R}}$ for the following expansion of L.

$$L_{\mathcal{R}} = (\Sigma_L \cup \{f_{\mathcal{R}}, c_{\mathcal{R}}\}, \mathcal{X}_L, \mathcal{F}_{\mathcal{R}}).$$

For each rigid equation $E \vdash_{\forall} s \approx t$ in $\mathcal{R}$, consider a fixed enumeration

$$\{ l_i \to r_i \mid 1 \leq i \leq |R_E| \} = R_E$$

and let $\vec{l} = l_1, l_2, \ldots, l_{|R_E|}$ and $\vec{r} = r_1, r_2, \ldots, r_{|R_E|}$. It is easy to check that $(L, c_{\mathcal{R}}, f_{\mathcal{R}}, F_{E \vdash_{\forall} s \approx t}, |R_E|, L_{\mathcal{R}}, s, t, \vec{l}, \vec{r})$ is appropriate. Define

$$SOE(\mathcal{R}, E \vdash_{\forall} s \approx t) = SOE(c_{\mathcal{R}}, f_{\mathcal{R}}, F_{E \vdash_{\forall} s \approx t}, s, t, \vec{l}, \vec{r}),$$

and

$$SOE(\mathcal{R}) = \{ SOE(\mathcal{R}, E \vdash_{\forall} s \approx t) \mid E \vdash_{\forall} s \approx t \in \mathcal{R} \}.$$

The construction corresponding to $SOE(\mathcal{R})$ is applied in Levy [22] to a variant of SREU called *simultaneous ground rigid O-unification* that disallows equations of the form $t \approx x$, where x is a variable, to appear in the left-hand side of rigid equations. The following example shows that the notion of guardedness is important also in this restricted case.

Example 1 Consider the system $\mathcal{R} = \{\emptyset \vdash_{\forall} x = y\}$ where x and y are variables. Then $SOE(\mathcal{R})$ has the form $\{F(f(x, c)) \approx f(y, F(c))\}$. Let $\theta = \{F \mapsto f(z_1, z_1), x \mapsto c, y \mapsto f(c, c)\}$. It is easy to check that θ solves $SOE(\mathcal{R})$ but it doesn't solve $\mathcal{R}$.

Theorem 2 *Let L be a first-order language and $\mathcal{R}$ a guarded system of rigid equations in L. The following statements are equivalent for all θ in $L_{\mathcal{R}}$ such that* $\mathrm{dom}(\theta) \cap \mathcal{F}_{\mathcal{R}} = \emptyset$.

(i) Some extension of θ with $\mathcal{F}_{\mathcal{R}}$ solves $SOE(\mathcal{R})$.

(ii) θ solves $\mathcal{R}$.

Proof. Let L, $\mathcal{R}$ and θ be given. Let $\mathcal{R} = \{ E_i \vdash_{\forall} s_i \approx t_i \mid 1 \leq i \leq n \}$ and let us write F_i for $F_{E_i \vdash_{\forall} s_i \approx t_i}$ and R_i for R_{E_i}. The following holds by Birkhoff's completeness theorem for all i, $1 \leq i \leq n$, assuming θ is grounding for $\mathcal{R}$:

$$\theta \text{ solves } E_i \vdash_{\forall} s_i \approx t_i \quad \Leftrightarrow \quad t_i\theta \xrightarrow{*}_{R_i\theta} s_i\theta. \tag{7}$$

((i)⇒(ii)) Assume that $\theta' = \theta \cup \{ F_i \mapsto t'_i \mid 1 \leq i \leq n \}$ solves $SOE(\mathcal{R})$.

First we show that, for $1 \leq i \leq n$, $R_i\theta$ is a set of ground rules in L and $s_i\theta$ is a ground term in L. Consider a fixed i. Let x be a variable in R_i or s_i. Let $E_j \vdash_\forall s_j \approx t_j$ be a guard for x in $\mathcal{R}$. So R_j and s_j are ground and $\theta \cup \{F_j \mapsto t'_j\}$ solves $SOE(\mathcal{R}, E_j \vdash_\forall s_j \approx t_j)$. Hence, it follows from Lemma 2((i)⇒(ii)) that $t_j\theta \in \mathcal{T}_{\Sigma_L}$. But x occurs in t_j, and thus $x\theta \in \mathcal{T}_{\Sigma_L}$.

Second, since $\theta \cup \{F_i \mapsto t'_i\}$ solves $SOE(\mathcal{R}, E_i \vdash_\forall s_i \approx t_i)$ and all the terms in $R_i\theta$ and $s_i\theta$ are in $\mathcal{T}_{\Sigma_L}$, it follows again from Lemma 2((i)⇒(ii)) that $t_i\theta \xrightarrow{*}_{R_i\theta} s_i\theta$. Thus, it follows from (7) that θ solves $E_i \vdash_\forall s_i \approx t_i$. Consequently θ solves $\mathcal{R}$.

((i)⇐(ii)) Assume that θ solves $\mathcal{R}$. From (7) and Lemma 2((i)⇐(ii)) follows that, for each i, $1 \leq i \leq n$, there is a term t'_i such that $\theta \cup \{F_i \mapsto t'_i\}$ solves $SOE(\mathcal{R}, E_i \vdash_\forall s_i \approx t_i)$. The rest is obvious, since $F_i \neq F_j$ for $i \neq j$. ⊠

Note that an analogous way to prove Theorem 2 when starting from an arbitrary system of of rigid equations $\mathcal{R}$ over a signature Σ, is to add the second-order equation corresponding to $Gr(\Sigma, x)$ (in Lemma 3), for each variable x in $\mathcal{R}$ (that has no guard in $\mathcal{R}$), to the resulting system of second-order equations.

Clearly, the system $SOE(\mathcal{R})$ is just another representation of $\mathcal{R}$. It follows from Theorem 2 that any semidecision algorithm for SOU that also produces a solution if one exists, can directly be used as a semidecision procedure for guarded SREU that also produces a solution if one exists.

Theorem 3 *SREU is logspace equivalent to SOU.*

Proof. By Lemma 3, Theorem 1, and Theorem 2. ⊠

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 [7, 10] 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 [12], whereas monadic SREU is only known to reduce to a nontrivial extension of word equations [19]. The decidability of monadic SREU is currently an open problem, only some special cases are known to be decidable [19, 6]. The decidability of monadic SREU is equivalent to the decidability of the prenex fragment of intuitionistic logic with equality restricted to function symbols of arity ≤ 1. A recent report by Voronkov discusses in detail the connections between SREU and other related problems [35].

Undecidability of a restricted case of SOU

The following theorem is a central result in Levy [22]. It is proved by reducing simultaneous ground rigid O-unification to SOU, by using a construction corresponding to $SOE(R)$ and noting that the reduction in Degtyarev and Voronkov [10] can be adapted to simultaneous ground rigid O-unification.

Theorem 4 (Levy [22]) *SOU is undecidable when restricted to systems of equations such that each second-order variable occurs at most twice in the system.*

By using the following lemma (an immediate corollary of [18, Theorem 18]) and Theorem 2, we can conclude that Theorem 4 holds already with very strong restrictions on the number of variables.

Lemma 4 (Gurevich-Veanes [18]) *Solvability of guarded systems of rigid equations with at most three rigid equations and at most two variables is undecidable.*

Theorem 5 *SOU is undecidable when restricted to systems $\mathcal{S}$ such that*

1. *each second-order variable occurs at most twice in $\mathcal{S}$,*
2. *there are at most 3 second-order variables in $\mathcal{S}$, and*
3. *there are at most 2 first-order variables in $\mathcal{S}$.*

Proof. Let $\mathcal{R}$ be a guarded system of three rigid equations with two variables. Then $SOE(\mathcal{R})$ is a system of second-order equations that satisfies the restrictions (1–3). By Theorem 2, $\mathcal{R}$ is solvable if and only if $SOE(\mathcal{R})$ is solvable. The undecidability follows from Lemma 4. $\boxtimes$

We can note that the second-order equations that we obtain in Theorem 5 are not simple (even if we consider the most restricted case in [18]). In order to prove the undecidability of the other cases of SOU that are listed in the introduction we have to apply the techniques that underlie the proof of Lemma 4 directly in the context of second-order unification. We do this in Section 5. We note, however, that the main result in Section 5 is an independent result that does not imply Theorem 5, because the number of variable occurrences is violated. For example, the decidability of SOU restricted to 2 second-order variables, each occurring at most twice, remains an open problem. This case is strongly related to the decidability of SREU with 2 rigid-equations, which is also an open problem.

5 A new elementary undecidability proof of SOU

In this section we present a new elementary undecidability proof of SOU, that does not rely on the deep theory underlying the undecidability proof of Hilbert's tenth problem due to Matiyasevich [24], that is used in Goldfarb [17]. As a corollary we can show the undecidability of SOU in more restricted cases than known previously. We adopt techniques from Gurevich and Veanes [18] and Veanes [30, 31], that originate from techniques in Plaisted [26], Goldfarb [17] and Hopcroft and Ullman [20, Lemma 8.6].

The undecidability proof is by reduction from the halting problem for Turing machines. We consider a fixed deterministic Turing machine M with *initial state* q_0, *final state* q_{f}, a *blank* character $\flat$, and an *input alphabet* that does not include the blank. 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. M is allowed to write blanks, however, M is only allowed to write a blank when it erases the *last* nonblank symbol on the tape and the tape head must move left after that. We assume, without loss of generality, that when M enters the final state then its tape is empty.

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 nonfinal; v^+ is the empty string (ϵ), otherwise (i.e., $q_{\mathrm{f}}^+ = \epsilon$).

5.1 Main idea

We construct two second-order equations from M: $S^M_{\mathrm{mv}}(F, G)$ and $S^M_{\mathrm{sp}}(x, F)$, that have roughly the following properties: (for any substitution θ such that $G \notin \mathrm{dom}(\theta)$ and input string v_0 for M)

1. $\theta \cup \{G \mapsto t\}$ solves $S^M_{\mathrm{mv}}(F, G)$ (for some t) 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 $S^M_{\mathrm{sp}}(q_0v_0, F)$ if and only if $F\theta$ represents the *shifted pairing* of a sequence of IDs $(v_1, v_2, \ldots, v_k)$ where $v_1 = q_0v_0$. (See Figure 1.)

Consequently, θ solves both second-order equations (for some $G\theta$) if and only if $F\theta$ represents a valid computation of M with input v_0.

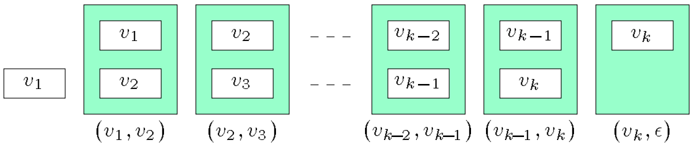

Figure 1: $((v_1, v_2), (v_2, v_3), \ldots, (v_k, \epsilon))$ is a *shifted pairing* of $(v_1, v_2, \ldots, v_k)$.

5.2 Encoding sequences of moves

We introduce a family of new constants $\{c_{ab}\}_{a,b \in \Sigma_M}$ 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 $v^+ = b_1 b_2 \cdots b_n$. (Note that $m - 1 \leq n \leq m + 1$.) We let $\langle v, v^+ \rangle$ denote the following string:

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

we call such a string a *move* also. (Note that $\langle q_{\mathrm{f}}, \epsilon \rangle = c_{q_{\mathrm{f}} \text{ƀ}}$.) 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}, and let Σ_{id} and Σ_{mv} be the following signatures:

$$\begin{aligned} \Sigma_{\mathrm{id}} &= \Sigma_M \cup \{c_{\mathrm{w}}, f_{\mathrm{w}}\}, \\ \Sigma_{\mathrm{mv}} &= \{\, c_{ab} \mid (a, b) \in \Sigma_M \times \Sigma_M \,\} \cup \{c_{\mathrm{w}}, f_{\mathrm{w}}, c_{\mathrm{t}}, f_{\mathrm{t}}\}. \end{aligned}$$

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 the corresponding string surrounded by double quotes:

$$f_{\mathrm{w}}(a_1, f_{\mathrm{w}}(a_2, \ldots, f_{\mathrm{w}}(a_n, c_{\mathrm{w}}) \cdots)) = \text{“}a_1 a_2 \cdots a_n\text{”},$$

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 moves is called a *move-train*. The following lemma is used together with Lemma 2 to construct the second-order equation $S^M_{\mathrm{mv}}(F, G)$ with the desired properties.

Lemma 5 *There is a system R_{mv} of ground rules of depth ≤ 1 over a signature Σ'_{mv}, where $\Sigma_{\mathrm{mv}} \subseteq \Sigma'_{\mathrm{mv}}$ and $\Sigma'_{\mathrm{mv}} \setminus \Sigma_{\mathrm{mv}}$ is a set of constants, such that, for all terms $t \in \mathcal{T}_{\Sigma_{\mathrm{mv}}}$, t is a move-train if and only if $t \xrightarrow{*}_{R_{\mathrm{mv}}} c_{\mathrm{t}}$.*

Proof. One can construct a deterministic finite bottom-up tree automaton (or DTA) that recognizes the set of all move-trains [18, 30, Train Theorem]. In particular, such a DTA can be constructed with one final state. When viewing DTAs as certain ground rewrite systems (see for example Dauchet [2]) then the rule set $R_{\rm mv}$ is simply the rule set of that DTA. $\boxtimes$

5.3 The main reduction

Throughout the rest of this section we use the following shorthand notation. Let $R_{\rm mv}$ and $\Sigma'_{\rm mv}$ be given by Lemma 5, such that $(\Sigma'_{\rm mv} \setminus \Sigma_{\rm mv}) \cap \Sigma_{\rm id} = \emptyset$.

- $m = |\Sigma_M|^2$ and $(a_1, b_1), (a_2, b_2), \ldots, (a_m, b_m)$ is a fixed sequence of all the pairs in $\Sigma_M \times \Sigma_M$.
- $\vec{c} = c_{a_1b_1}, c_{a_2b_2}, \ldots, c_{a_mb_m}$, note that $\vec{c} \in \mathcal{T}_{\Sigma_{\rm mv}}$.
- $\vec{a} = a_1, a_2, \ldots, a_m$ and $\vec{l}_{\rm sp} = \vec{a}$, "$\not b$", $c_{\rm w}$, "$\not b$", note that $\vec{l}_{\rm sp} \in \mathcal{T}_{\Sigma_{\rm id}}$.
- $\vec{b} = b_1, b_2, \ldots, b_m$ and $\vec{r}_{\rm sp} = \vec{b}, c_{\rm w}$, "$\not b$", "$\not b$", note that $\vec{r}_{\rm sp} \in \mathcal{T}_{\Sigma_{\rm id}}$.
- $m_1 = m + 3$ and F is a new second-order variable with arity $m_1 + 1$.
- Consider a fixed sequence of all the rules in $R_{\rm mv}$. Let $\vec{l}_{\rm mv}$ be the corresponding sequence of all the left-hand sides, and $\vec{r}_{\rm mv}$ the corresponding sequence of all the right-hand sides. Note that $\vec{l}_{\rm mv}, \vec{r}_{\rm mv} \in \mathcal{T}_{\Sigma'_{\rm mv}}$.
- $m_2 = |R_{\rm mv}|$ and G is a new second-order variable with arity $m_2 + 1$.
- $L_{\rm id} = (\Sigma_{\rm id}, \emptyset, \emptyset)$.
- $L'_{\rm mv} = (\Sigma'_{\rm mv}, \emptyset, \{F\})$.
- d is a new constant and g is a new binary function symbol.
- L is (any expansion of) $(\Sigma_{\rm id} \cup \Sigma'_{\rm mv} \cup \{d, g\}, \emptyset, \{F, G\})$.

We use the following facts without further notice:

- $(L_{\rm id}, c_{\rm t}, f_{\rm t}, F, m_1, L,$ "$\not b$"$, t, \vec{l}_{\rm sp}, \vec{r}_{\rm sp})$ (for $t \in \mathcal{T}_{\Sigma_{\rm id}}$) is appropriate.
- $(L'_{\rm mv}, d, g, G, m_2, L, c_{\rm t}, F(\vec{c}, c_{\rm w}, c_{\rm w}, c_{\rm w}, c_{\rm t}), \vec{l}_{\rm mv}, \vec{r}_{\rm mv})$ is appropriate.

For $t \in \mathcal{T}_\Sigma$, we define the second-order equations $S^M_{\mathrm{sp}}(t,F)$ and $S^M_{\mathrm{mv}}(F,G)$ and the system $S^M(t,F,G)$ in L as follows:

$$\begin{array}{rcl} S^M_{\mathrm{sp}}(t,F) & = & SOE(c_{\mathrm{t}}, f_{\mathrm{t}}, F, \text{“}\not b\text{”}, t, \vec{l}_{\mathrm{sp}}, \vec{r}_{\mathrm{sp}}), \\ S^M_{\mathrm{mv}}(F,G) & = & SOE(d, g, G, c_{\mathrm{t}}, F(\vec{c}, c_{\mathrm{w}}, c_{\mathrm{w}}, c_{\mathrm{w}}, c_{\mathrm{t}}), \vec{l}_{\mathrm{mv}}, \vec{r}_{\mathrm{mv}}), \\ S^M(t,F,G) & = & \{S^M_{\mathrm{sp}}(t,F), S^M_{\mathrm{mv}}(F,G)\}. \end{array}$$

Let us briefly recall the intuition behind this construction. Assume that θ solves the system. To start with consider $S^M_{\mathrm{mv}}(F\theta, G\theta)$. It follows from Lemma 5 (with a little help from Lemma 6 below) that $F\theta[\vec{c}, c_{\mathrm{w}}, c_{\mathrm{w}}, c_{\mathrm{w}}, c_{\mathrm{t}}]$ is a term in $\mathcal{T}_{\Sigma_{\mathrm{mv}}}$ representing a sequence of moves:

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

Next consider $S^M_{\mathrm{sp}}(t, F\theta)$ and Lemma 1, which tells us firstly that the function symbols in $F\theta$ are in $\Sigma_{\mathrm{id}} \cup \{f_{\mathrm{t}}\}$. So $F\theta$ cannot contain any symbols from $\vec{c}$, because the only constant that Σ_{id} and Σ_{mv} have in common is the empty word. Consequently, $F\theta$ has the bound variable z_i for every occurrence of $c_{a_i b_i}$ in $F\theta[\vec{c}, c_{\mathrm{w}}, c_{\mathrm{w}}, c_{\mathrm{w}}, c_{\mathrm{t}}]$. Therefore $F\theta[\vec{l}_{\mathrm{sp}}, f_{\mathrm{t}}(\text{“}\not b\text{”}, c_{\mathrm{t}})]$ represents (roughly) the sequence:

$$(v_1, v_2, \ldots, v_k, \not b),$$

and $F\theta[\vec{r}_{\mathrm{sp}}, c_{\mathrm{t}}]$ represents (roughly) the sequence:

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

and the conditions (2–4) in Lemma 1 imply that $(v_1, v_2, \ldots, v_k)$ is a valid computation.

Lemma 6 *Let $t \in \mathcal{T}_{\Sigma_{\mathrm{id}}}$. If θ solves $S^M(t,F,G)$, then $F\theta[\vec{c}, c_{\mathrm{w}}, c_{\mathrm{w}}, c_{\mathrm{w}}, c_{\mathrm{t}}] \in \mathcal{T}_{\Sigma_{\mathrm{mv}}}$.*

Proof. Given $t \in \mathcal{T}_{\Sigma_{\mathrm{id}}}$ and θ in L, assume that θ solves $S^M(t,F,G)$. Then θ solves $S^M_{\mathrm{sp}}(t,F)$. It follows from Lemma 1 that $F\theta$ is a closed $(\Sigma_{\mathrm{id}} \cup \{f_{\mathrm{t}}\}, \emptyset)^*$-term of rank $\leq m_1 + 1$. Since θ solves also $S^M_{\mathrm{mv}}(F,G)$, Lemma 2 implies that $F\theta[\vec{c}, c_{\mathrm{w}}, c_{\mathrm{w}}, c_{\mathrm{w}}, c_{\mathrm{t}}] \in \mathcal{T}_{\Sigma'_{\mathrm{mv}}}$. The rest is obvious from the fact that $(\Sigma'_{\mathrm{mv}} \setminus \Sigma_{\mathrm{mv}}) \cap \Sigma_{\mathrm{id}} = \emptyset$. $\boxtimes$

Given an input string v for M, we let $S^M_v(F,G) = S^M(\text{“}q_0 v\text{”}, F, G)$. (Recall that $q_0 v$ is the initial ID of M with input v.) We can now prove the main theorem.

Theorem 6 *For any input string v_0 for M, $S^M_{v_0}(F,G)$ is solvable if and only if M accepts v_0.*

Proof. Let v_0 be an input string for M.

($\Rightarrow$) Assume that $S^M_{v_0}(F,G)$ is solvable. Let θ be a substitution in L that solves $S^M_{v_0}(F,G)$. Since θ solves $S^M_{\rm mv}(F,G)$, it follows from Lemma 2 that

$$F\theta[\vec{c}, c_{\rm w}, c_{\rm w}, c_{\rm w}, c_{\rm t}] \xrightarrow{*}_{R_{\rm mv}} c_{\rm t}.$$

Hence, by Lemma 6 and Lemma 5, $F\theta[\vec{c}, c_{\rm w}, c_{\rm w}, c_{\rm w}, c_{\rm t}]$ is a move-train:

$$\begin{aligned} &F\theta[\vec{c}, c_{\rm w}, c_{\rm w}, c_{\rm w}, c_{\rm t}] = \\ &\quad f_{\rm t}(\text{“}\langle v_1, v_1^+\rangle\text{”}, f_{\rm t}(\text{“}\langle v_2, v_2^+\rangle\text{”}, \ldots, f_{\rm t}(\text{“}\langle v_k, v_k^+\rangle\text{”}, c_{\rm t})\cdots)), \end{aligned} \tag{8}$$

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

$$F\theta = f_{\rm t}(s_1, f_{\rm t}(s_2, \ldots, f_{\rm t}(s_k, z_{m_1+1})\cdots)), \tag{9}$$

where each s_i is a closed $L^*_{\rm id}$-term of rank $\leq m_1$ and $k \geq 1$. (The case $F\theta = z_{m_1+1}$, i.e., $k=0$, is not possible because “q_0v_0” $\neq$ “$\not b$”.) Moreover,

$$\begin{aligned} \text{“}q_0v_0\text{”} &= s_1[\vec{l}_{\rm sp}], & & (10)\\ s_i[\vec{r}_{\rm sp}] &= s_{i+1}[\vec{l}_{\rm sp}], \quad \text{for } 1 \leq i < k, & & (11)\\ s_k[\vec{r}_{\rm sp}] &= \text{“}\not b\text{”}. & & (12) \end{aligned}$$

It follows from (8), (9), and each s_i having rank $\leq m_1$, that

$$s_i[\vec{c}, c_{\rm w}, c_{\rm w}, c_{\rm w}] = \text{“}\langle v_i, v_i^+\rangle\text{”}, \quad \text{for } 1 \leq i \leq k. \tag{13}$$

Note that $s_i[\vec{c}, c_{\rm w}, c_{\rm w}, c_{\rm w}] \in \mathcal{T}_{\Sigma_{\rm mv}}$ and $s_i[\vec{l}_{\rm sp}], s_i[\vec{r}_{\rm sp}] \in \mathcal{T}_{\Sigma_{\rm id}}$ for $1 \leq i \leq k$. The only constant that can occur in any s_i is therefore $c_{\rm w}$.

In order to show that M accepts v_0 we show that $v_1 = q_0v_0$, $v_i^+ = v_{i+1}$ for $1 \leq i < k$, and $v_k = q_{\rm f}$. Consider a fixed i, $1 \leq i \leq k$. Recall that the j'th constant in the sequence $\vec{c}$ is $c_{a_jb_j}$ and it occurs only at position j in $\vec{c}$ (since there are no two identical constants in $\vec{c}$). Thus, given that

$$\text{“}\langle v_i, v_i^+\rangle\text{”} = f_{\rm w}(c_{a_{j_1}b_{j_1}}, f_{\rm w}(c_{a_{j_2}b_{j_2}}, \ldots, f_{\rm w}(c_{a_{j_n}b_{j_n}}, c_{\rm w})\cdots)),$$

it follows from (13) that

$$s_i = f_{\rm w}(z_{j_1}, f_{\rm w}(z_{j_2}, \ldots, f_{\rm w}(z_{j_n}, s_i')\cdots)),$$

where $s_i' \in \{c_{\rm w}, z_{m_1-2}, z_{m_1-1}, z_{m_1}\}$, and $1 \leq j_{n'} \leq m$ for $1 \leq n' \leq n$. Recall that, for $j \leq m$, the j'th constant in $\vec{l}_{\rm sp}$ is a_j and the j'th constant in $\vec{r}_{\rm sp}$ is b_j. Hence, for $1 \leq i \leq k$,

$$\begin{aligned} s_i[\vec{l}_{\rm sp}] &\in \{\text{“}v_i\text{”}, \text{“}v_i\not b\text{”}, \text{“}v_i\not b\not b\text{”}\},\\ s_i[\vec{r}_{\rm sp}] &\in \{\text{“}v_i^+\text{”}, \text{“}v_i^+\not b\text{”}, \text{“}v_i^+\not b\not b\text{”}\}. \end{aligned}$$

Now, (10) implies that $v_1 = q_0v_0$, (11) implies that $v_i^+ = v_{i+1}$ for $1 \leq i < k$, and (12) implies that “$v_k^+\not b$” $=$ “$\not b$”, i.e., $v_k^+ = \epsilon$ and hence $v_k = q_{\rm f}$. Thus M accepts v_0.

($\Leftarrow$) Assume that M accepts v_0. We construct a substitution θ that solves $S_{v_0}^M(F, G)$. Consider a valid computation of M with input v_0:

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

for some $k \geq 1$, i.e., $v_1 = q_0 v_0$, $v_i^+ = v_{i+1}$ for $1 \leq i \leq k$, and $v_{k+1} = \epsilon$ ($v_k = q_{\mathrm{f}}$). We construct $F\theta$ like above, where the s_i's are the following terms. Let i, $1 \leq i \leq k$, be fixed and assume that

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

where $1 \leq j_{n'} \leq m$ for $1 \leq n' \leq n$. Let

$$s_i = f_{\mathrm{w}}(z_{j_1}, f_{\mathrm{w}}(z_{j_2}, \ldots, f_{\mathrm{w}}(z_{j_n}, s_i') \cdots)),$$

where s_i' is one of c_{w}, z_{m_1-2}, z_{m_1-1} or z_{m_1} (specified below). Given such s_i for $1 \leq i \leq k$, obviously

$$s_i[\vec{c}, c_{\mathrm{w}}, c_{\mathrm{w}}, c_{\mathrm{w}}] = \text{“}\langle v_i, v_{i+1} \rangle\text{”}, \quad (1 \leq i \leq k).$$

Hence, $F\theta[\vec{c}, c_{\mathrm{w}}, c_{\mathrm{w}}, c_{\mathrm{w}}, c_{\mathrm{t}}]$ is a move-train. It follows from Lemma 5 that

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

and then from Lemma 2((ii)$\Rightarrow$(i)) that (for some term $G\theta$) θ solves $S_{\mathrm{mv}}^M(F, G)$.

Next, we choose the s_i''s, for $1 \leq i \leq k$, in $\{c_{\mathrm{w}}, z_{m_1-2}, z_{m_1-1}, z_{m_1}\}$, so that the conditions (10–12) hold. Consider $\langle v_i, v_{i+1} \rangle$ above, let us call $a_{j_1} a_{j_2} \cdots a_{j_n}$ its *first projection*, denoted by $\pi_1(\langle v_i, v_{i+1} \rangle)$, and $b_{j_1} b_{j_2} \cdots b_{j_n}$ its *second projection*, denoted by $\pi_2(\langle v_i, v_{i+1} \rangle)$. It is easy to check that for a given choice of s_i', the terms $s_i[\vec{l}_{\mathrm{sp}}]$ and $s_i[\vec{r}_{\mathrm{sp}}]$ are as follows:

$s_i' =$	$s_i[\vec{l}_{\mathrm{sp}}] =$	$s_i[\vec{r}_{\mathrm{sp}}] =$
c_{w}	“$\pi_1(\langle v_i, v_{i+1} \rangle)$”	“$\pi_2(\langle v_i, v_{i+1} \rangle)$”
z_{m_1-2}	“$\pi_1(\langle v_i, v_{i+1} \rangle)\not{b}$”	“$\pi_2(\langle v_i, v_{i+1} \rangle)$”
z_{m_1-1}	“$\pi_1(\langle v_i, v_{i+1} \rangle)$”	“$\pi_2(\langle v_i, v_{i+1} \rangle)\not{b}$”
z_{m_1}	“$\pi_1(\langle v_i, v_{i+1} \rangle)\not{b}$”	“$\pi_2(\langle v_i, v_{i+1} \rangle)\not{b}$”

Now, the key point to observe is that, for $1 < i \leq k$, if $\pi_2(\langle v_{i-1}, v_i \rangle)$ and $\pi_1(\langle v_i, v_{i+1} \rangle)$ are distinct, the one of them is v_i and the other one is $v_i\not{b}$. Hence, the table shows that it is possible to define the s_i''s so that $s_i[\vec{r}_{\mathrm{sp}}] = s_{i+1}[\vec{l}_{\mathrm{sp}}]$ for $1 \leq i < k$ and $s_k[\vec{r}_{\mathrm{sp}}] = $ “$\not{b}$”. It is also necessary to assume that $\pi_1(\langle v_1, v_2 \rangle) = v_1$, and to define s_1' so that $s_1[\vec{l}_{\mathrm{sp}}] = $ “$q_0 v_0$”. The conditions (10–12) follow. Hence, θ solves S_{sp}^M(“$q_0 v_0$”$, F)$ by Lemma 1((ii)$\Rightarrow$(i)). $\boxtimes$

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 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 and Ullman [20]. The universal TM accepts $\langle M, v\rangle$ if and only if M accepts v. The following corollary is an easy consequence of Theorem 6. Recall that a *simple* second-order equation is one where all occurrences of second-order variables have ground arguments.

Corollary 1 *There is a simple second-order equation $S^{\mathrm{u}}(x, F, G)$ of depth 4 and of the form*

$$g(F(\vec{t}_1), G(\vec{t}_2)) \approx g(f(x, F(\vec{t}_3)), g(F(\vec{t}_4), G(\vec{t}_5))),$$

such that the problem of determining whether $S^{\mathrm{u}}(t, F, G)$ is solvable for a given ground term t, is undecidable.

Proof. Consider the system

$$S^{M_{\mathrm{u}}}(x, F, G) = \{F(\vec{t}_1) \approx f(x, F(\vec{t}_3)),\ G(\vec{t}_2) \approx g(F(\vec{t}_4), G(\vec{t}_5))\},$$

for some sequences $\vec{t}_i$, $1 \leq i \leq 5$, of ground terms. Pair the two equations together to form $S^{\mathrm{u}}(x, F, G)$. The depth of $S^{\mathrm{u}}(x, F, G)$ is 4 since the elements of each $\vec{t}_i$ have depth ≤ 1. Evidently, for any given term t, $S^{\mathrm{u}}(t, F, G)$ is solvable if and only if $S^{M_{\mathrm{u}}}(t, F, G)$ is solvable. In particular, given a TM M and input v for M, $S^{\mathrm{u}}(\text{“}q_0\langle M, v\rangle\text{”}, F, G)$ is solvable if and only if $S^{M_{\mathrm{u}}}_{\langle M,v\rangle}(F, G)$ is solvable if and only if (by Theorem 6) M_{u} accepts $\langle M, v\rangle$ if and only if M accepts v. ⊠

Let us call the second-order equation S^{u} in Corollary 1 a *universal second-order equation* and let us denote the language of S^{u} by L_{u}.

Corollary 2 *Second-order unification is undecidable under the following restrictions:*

1. *there are no first-order variables,*
2. *at most two variables,*
3. *at most five occurrences of variables,*
4. *the equations are simple (Schubert [28]), and*
5. *the arguments of all variables have constantly bounded depth.*

In the following section we get further improvements of this result by applying certain encoding techniques in Farmer [13].

It is interesting to note that, by applying the reduction in Degtyarev and Voronkov [10] to a system of *simple* second-order equations, one obtains a system of rigid equations with *ground left-hand sides.*[5] (The converse does not hold, i.e., the reduction from a system of rigid equations with ground left-hand sides, by using Theorem 2, does in general not yield a system of simple second-order equations.) Thus, by using Schubert's result that is confirmed by Corollary 2, one obtains an elegant proof of the following statement.

Corollary 3 (Plaisted [26]) *SREU is undecidable with ground left-hand sides.*

6 A complement to Farmer's theorem

Let us recall the following result.

Theorem 7 (Farmer [13]) *There is an integer n such that the second-order unification problem is undecidable for all nonmonadic languages, that contain at least n second-order variables.*

One important point of this result is that all the second-order variables may be unary. So undecidability of second-order unification arises for all non-monadic languages with sufficiently many second-order variables, even if all of them are unary and there are no first-order variables and only one constant in the language. However, as is noted in Farmer [13, page 30], *there is a possibility, that there is some second-order, nonmonadic language containing a small number of second-order variables for which second-order unification is decidable.*

The main result of this section is Theorem 8, showing that this is not possible if the *arities* of the second-order variables are large enough, in which case the undecidability arises already with two second-order variables. The above possibility remains only if there is one second-order variable, or if there is a small number of second-order variables with low arities.

We use a special case of a result in Farmer [13, Lemma 6.5] (Lemma 7) to show that Corollary 2 holds for all nonmonadic languages. We make explicit some additional information that we extract from the main part of its proof.[6]

[5] This observation was made by Voronkov.

[6] Lemma 7 holds actually not just for simple equations but for a larger class of second-order equations that are called "rigid" by Farmer. This notion is not related to the definition of rigid equations in the context of rigid E-unification.

Lemma 7 *Let L_1 be a nonmonadic second-order language and let L be the language $(\{f, c\}, \mathcal{X}_{L_1}, \mathcal{F}_{L_1})$, where f is a binary function symbol and c is a constant. There is an effective mapping $\varphi : \mathcal{T}_{L_1} \to \mathcal{T}_L$ such that, for all simple $e = s \approx t$ in L_1:*

1. *$\varphi(e) = \varphi(s) \approx \varphi(t)$ in L is simple,*
2. *e is solvable in L_1 if and only if $\varphi(e)$ is solvable in L,*
3. *the set of variables in e coincides with the set of variables in $\varphi(e)$,*
4. *the number of variable occurrences in e and $\varphi(e)$ are equal,*
5. *if $s = s_1\{x \mapsto s_2\}$ for some simple s_1 and ground s_2 then $\varphi(s) = \varphi(s_1)\{x \mapsto \varphi(s_2)\}$.*

Theorem 8 *There is a positive integer n, such that, for any nonmonadic second-order language L with at least two second-order variables with arity $\geq n$, the unification problem for L is undecidable already for simple equations with at most five variable occurrences having arguments of depth $\leq n$.*

Proof. It is enough to prove the statement for some second-order variables F and G and $L = (\{f, c\}, \emptyset, \{F, G\})$, where f is a binary function symbol, and c is a constant (cf [13, Lemma 2.1]). Let L' be L expanded with the first-order variable x. Consider the universal second-order equation $S^{\mathrm{u}}(x, F, G)$ (that is obviously simple) and let $\varphi : \mathcal{T}_{L_{\mathrm{u}}} \to \mathcal{T}_{L'}$ be given by Lemma 7. Let $S'(x, F, G) = \varphi(S^{\mathrm{u}}(x, F, G))$. It follows that for any term $t \in \mathcal{T}_{\Sigma_{L_{\mathrm{u}}}}$, $S^{\mathrm{u}}(t, F, G)$ is solvable in L_{u} if and only if $\varphi(S^{\mathrm{u}}(t, F, G))$ is solvable in L' (i.e., in L) if and only if $S'(\varphi(t), F, G)$ is solvable in L. The statement follows now from Corollary 1 for n equal to some integer greater that the arities of F and G and the depth of $S'(x, F, G)$. ⊠

7 Some open problems

Despite the similarity of SREU and second-order unification in general, their monadic fragments (i.e., when all function symbols have arity ≤ 1) seem to be farther apart. The reason is that the decidability of monadic second-order unification can be proved by reduction to word equations [12], whereas monadic SREU is only known to reduce to a nontrivial extension of word equations [19] and its decidability is currently an open problem, with only some special decidable cases [19, 6]. It is also known that SREU is undecidable with three rigid equations [18] and decidable with one rigid equation [14].

The two rigid equations case remains an intriguing open problem, and the relationship to second-order unification might be useful to settle this question. Further open problems related to SREU are discussed in Voronkov [34]. With respect to the number of variables, the decidability of SREU has recently been settled completely [4, 31].

We conjecture that, by applying the techniques that are used by Farmer (in particular [13, Lemma 5.2 and Lemma 6.1]), to the universal second-order equation S^{u}, one can obtain a more elementary proof of Farmer's main theorem that holds already for simple equations. We estimate that the number of unary second-order variables that the reduction from S^{u} would lead to is roughly $4n$ where n is the maximum of the arities of the second-order variables in S^{u}. It is still not known whether second-order unification is decidable for "small" number of second-order variables with "low" arities. As we have shown, when the arities can be large enough, then undecidability arises already with two second-order variables. It is also an open problem if second-order unification is decidable with one second-order variable, unless there are at most two occurrences of the second-order variable [22].

References

[1] H.P. Barendregt. Lambda calculi with types. In S. Abramsky, D.M. Gabbay, and T.S.E. Mainbaum, editors, *Handbook of Logic in Computer Science*, volume 2, pages 117–309. Oxford University Press, 1992.

[2] M. Dauchet. Rewriting and tree automata. In H. Comon and J.P. Jouannaud, editors, *Term Rewriting (French Spring School of Theoretical Computer Science)*, volume 909 of *Lecture Notes in Computer Science*, pages 95–113. Springer Verlag, Font Romeux, France, 1993.

[3] A. Degtyarev, Yu. Gurevich, P. Narendran, M. Veanes, and A. Voronkov. Decidability and complexity of simultaneous rigid E-unification with one variable and related results. *Theoretical Computer Science*, 1998. To appear.

[4] A. Degtyarev, Yu. Gurevich, P. Narendran, M. Veanes, and A. Voronkov. The decidability of simultaneous rigid E-unification with one variable. In *Rewriting Techniques and Applications*, 1998. To appear, also available as UPMAIL Technical Report 139, March 1997, Uppsala University, Computing Science Department.

[5] A. Degtyarev, Yu. Gurevich, and A. Voronkov. Herbrand's theorem and equational reasoning: Problems and solutions. UPMAIL Techni-

cal Report 128, Uppsala University, Computing Science Department, September 1996. Appears in the Bulletin of the European Association for Theoretical Computer Science (Vol 60, October 1996).

[6] A. Degtyarev, Yu. Matiyasevich, and A. Voronkov. Simultaneous rigid E-unification and related algorithmic problems. In *Eleventh Annual IEEE Symposium on Logic in Computer Science (LICS'96)*, pages 494–502, New Brunswick, NJ, July 1996. IEEE Computer Society Press.

[7] A. Degtyarev and A. Voronkov. Reduction of second-order unification to simultaneous rigid E-unification. UPMAIL Technical Report 109, Uppsala University, Computing Science Department, June 1995.

[8] A. Degtyarev and A. Voronkov. Simultaneous rigid E-unification is undecidable. UPMAIL Technical Report 105, Uppsala University, Computing Science Department, May 1995.

[9] A. Degtyarev and A. Voronkov. Decidability problems for the prenex fragment of intuitionistic logic. In *Eleventh Annual IEEE Symposium on Logic in Computer Science (LICS'96)*, pages 503–512, New Brunswick, NJ, July 1996. IEEE Computer Society Press.

[10] A. Degtyarev and A. Voronkov. The undecidability of simultaneous rigid E-unification. *Theoretical Computer Science*, 166(1–2):291–300, 1996.

[11] N. Dershowitz and J.-P. Jouannaud. Rewrite systems. In J. Van Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume B: Formal Methods and Semantics, chapter 6, pages 243–309. North Holland, Amsterdam, 1990.

[12] W.M. Farmer. A unification algorithm for second-order monadic terms. *Annals of Pure and Applied Logic*, 39:131–174, 1988.

[13] W.M. Farmer. Simple second-order languages for which unification is undecidable. *Theoretical Computer Science*, 87:25–41, 1991.

[14] J.H. Gallier, P. Narendran, D. Plaisted, and W. Snyder. Rigid E-unification is NP-complete. In *Proc. IEEE Conference on Logic in Computer Science (LICS)*, pages 338–346. IEEE Computer Society Press, July 1988.

[15] J.H. Gallier, S. Raatz, and W. Snyder. Theorem proving using rigid E-unification: Equational matings. In *Proc. IEEE Conference on Logic in Computer Science (LICS)*, pages 338–346. IEEE Computer Society Press, 1987.

[16] F. Gécseg and M. Steinby. *Tree Automata.* Akadémiai Kiodó, Budapest, 1984.

[17] W.D. Goldfarb. The undecidability of the second-order unification problem. *Theoretical Computer Science*, 13:225–230, 1981.

[18] Y. Gurevich and M. Veanes. Some undecidable problems related to the Herbrand theorem. UPMAIL Technical Report 138, Uppsala University, Computing Science Department, March 1997. Submitted to *Information and Computation.*

[19] Y. Gurevich and A. Voronkov. The monadic case of simultaneous rigid E-unification. In *ICALP'97*, Lecture Notes in Computer Science, 1997.

[20] J. E. Hopcroft and J. D. Ullman. *Introduction to Automata Theory, Languages and Computation.* Addison-Wesley Publishing Co., 1979.

[21] J. Levy. Linear second-order unification. In *Rewriting Techniques and Applications*, volume 1103 of *Lecture Notes in Computer Science*, pages 332–346. Springer Verlag, 1996.

[22] J. Levy. Decidable and undecidable second-order unification problems. Accepted for *RTA'98*, 1997.

[23] G.S. Makanin. The problem of solvability of equations in free semigroups. *Mat. Sbornik (in Russian)*, 103(2):147–236, 1977. English Translation in American Mathematical Soc. Translations (2), vol. 117, 1981.

[24] Yu.V. Matiyasevič. The diophantiness of recursively enumerable sets (in Russian). *Soviet Mathematical Doklady*, pages 279–282, 1970.

[25] U. Petermann. A complete connection calculus with rigid E-unification. In *JELIA'94*, volume 838 of *Lecture Notes in Computer Science*, pages 152–166, 1994.

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

[27] M. Schmidt-Schauß. Unification of stratified second-order terms. Technical Report 12/94, Johan Wolfgang-Göthe-Universität, Frankfurt, 1995.

[28] A. Schubert. Second-order unification and type inference for Church-style polymorphism. Technical report, Institute of Informatics, Warsaw University, January 1997. To appear in *POPL'98.*

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

[30] M. Veanes. *On Simultaneous Rigid E-Unification.* PhD thesis, Computing Science Department, Uppsala University, 1997.

[31] M. Veanes. The undecidability of simultaneous rigid E-unification with two variables. In *Proc. Kurt Gödel Colloquium KGC'97*, volume 1289 of *Lecture Notes in Computer Science*, pages 305–318. Springer Verlag, 1997.

[32] A. Voronkov. Proof search in intuitionistic logic based on constraint satisfaction. In P. Miglioli, U. Moscato, D. Mundici, and M. Ornaghi, editors, *Theorem Proving with Analytic Tableaux and Related Methods. 5th International Workshop, TABLEAUX '96*, volume 1071 of *Lecture Notes in Artificial Intelligence*, pages 312–329, Terrasini, Palermo Italy, May 1996.

[33] A. Voronkov. Proof search in intuitionistic logic with equality, or back to simultaneous rigid E-unification. In M.A. McRobbie and J.K. Slaney, editors, *Automated Deduction — CADE-13*, volume 1104 of *Lecture Notes in Computer Science*, pages 32–46, New Brunswick, NJ, USA, 1996.

[34] A. Voronkov. Herbrand's theorem, automated reasoning and semantic tableaux. UPMAIL Technical Report 151, Uppsala University, Computing Science Department, February 1998. Accepted for *LICS'98*.

[35] A. Voronkov. Simultaneous rigid e-unification and other decision problems related to the Herbrand theorem. UPMAIL Technical Report 152, Uppsala University, Computing Science Department, February 1998. Submitted to *TCS*.

INFORMATIK

Below you find a list of the most recent technical reports of the research group *Logic of Programming* at the Max-Planck-Institut für Informatik. They are available by anonymous ftp from our ftp server `ftp.mpi-sb.mpg.de` under the directory `pub/papers/reports`. Most of the reports are also accessible via WWW using the URL `http://www.mpi-sb.mpg.de`. If you have any questions concerning ftp or WWW access, please contact `reports@mpi-sb.mpg.de`. Paper copies (which are not necessarily free of charge) can be ordered either by regular mail or by e-mail at the address below.

Max-Planck-Institut für Informatik
Library
attn. Birgit Hofmann
Im Stadtwald
D-66123 Saarbrücken
GERMANY
e-mail: `library@mpi-sb.mpg.de`

MPI-I-98-2-006	P. Blackburn, M. Tzakova	Hybrid Languages and Temporal Logic
MPI-I-98-2-005	M. Veanes	The Relation Between Second-Order Unification and Simultaneous Rigid E-Unification
MPI-I-98-2-004	S. Vorobyov	Satisfiability of Functional+Record Subtype Constraints is NP-Hard
MPI-I-97-2-012	L. Bachmair, H. Ganzinger, A. Voronkov	Elimination of Equality via Transformation with Ordering Constraints
MPI-I-97-2-011	L. Bachmair, H. Ganzinger	Strict Basic Superposition and Chaining
MPI-I-97-2-010	S. Vorobyov, A. Voronkov	Complexity of Nonrecursive Logic Programs with Complex Values
MPI-I-97-2-009	A. Bockmayr, F. Eisenbrand	On the Chvátal Rank of Polytopes in the 0/1 Cube
MPI-I-97-2-008	A. Bockmayr, T. Kasper	A Unifying Framework for Integer and Finite Domain Constraint Programming
MPI-I-97-2-007	P. Blackburn, M. Tzakova	Two Hybrid Logics
MPI-I-97-2-006	S. Vorobyov	Third-order matching in $\lambda \rightarrow$-Curry is undecidable
MPI-I-97-2-005	L. Bachmair, H. Ganzinger	A Theory of Resolution
MPI-I-97-2-004	W. Charatonik, A. Podelski	Solving set constraints for greatest models
MPI-I-97-2-003	U. Hustadt, R.A. Schmidt	On evaluating decision procedures for modal logic
MPI-I-97-2-002	R.A. Schmidt	Resolution is a decision procedure for many propositional modal logics
MPI-I-97-2-001	D.A. Basin, S. Matthews, L. Viganò	Labelled modal logics: quantifiers
MPI-I-96-2-010	A. Nonnengart	Strong Skolemization
MPI-I-96-2-009	D.A. Basin, N. Klarlund	Beyond the Finite in Automatic Hardware Verification
MPI-I-96-2-008	S. Vorobyov	On the decision complexity of the bounded theories of trees
MPI-I-96-2-007	A. Herzig	SCAN and Systems of Conditional Logic