On Unification Problems in Restricted Second-Order Languages (Extended Abstract)

Jordi Levy[1] and Margus Veanes[2]

[1] Institut d'Investigació en Intel·ligència Artificial
Consejo Superior de Investigaciones Científicas
[2] Max-Planck-Institut für Informatik
Im Stadtwald, 66123 Saarbrücken, Germany

Abstract. We review known results and improve known boundaries between the decidable and the undecidable cases of second-order unification with various restrictions on second-order variables. As a key tool we prove an undecidability result that provides a partial solution to an open problem about simultaneous rigid E-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

NB We have added an appendix, that is not part of the extended abstract, to provide direct access to a complete proof of Theorem 7.

1 Introduction

Second-order unification and restricted forms of it play a fundamental role in several areas of computer science. Context unification [1, 20] and linear second-order unification [14] (a generalization of the former) are restricted forms of second-order unification that generalize the word unification problem [17]. Context unification appears as a subproblem in constraint solving with membership constraints [1], distributive unification [21] and completion of bi-rewriting systems [13]. The decidability of context unification is a difficult open problem, there has been some progress though, towards a decidability result [22].

Recently, a natural reduction from simultaneous rigid E-unification was found to second-order unification [15, 26], complementing the previously known converse reduction [4], and implying that the problems are in principal the same. Due to the fundamental role of simultaneous rigid E-unification in logic with equality [9, 28], second-order unification turns out to have close connections to Herbrand's theorem [28] and to intuitionistic logic with equality [27]. These relations shed a new light on the foundational aspects of automated reasoning in logic with equality.

The undecidability result of second-order unification [10], has led to investigations where certain decidable [1, 6, 14, 15, 18, 20] and undecidable [7, 15, 23, 26] subcases of second-order unification have been classified. Some recent progress in this matter has been made by using the connection to simultaneous rigid E-unification [15, 26]. The aim of this paper is to review the known results and to solve some open problems posed in [15, 26] in order to provide precise boundaries between the decidable and the undecidable cases of second-order unification with restrictions on the *number*, *the number of occurrences*, and the *arity* of second-order variables.

The rest of the paper is organized as follows. In Section 4 we prove that second-order unification reduces to second-order unification with one second-order variable, obtaining the following result:

- *Second-order unification is undecidable with one unary second-order variable.*

In Section 5 we obtain further undecidability results of some very restricted fragments of second-order unification, by using the following statement and results in [15, 26]. (A complete proof is given in the appendix.)

- *There are two ground rewrite systems R_1 and R_2 such that R_1 is canonical, R_2 is noetherian, and the following decision problem is undecidable.*
  - **Given***: first-order terms s_1, t_1, s_2, t_2, where s_1 is ground and all variables in s_2 or t_2 occur in t_1.*
  - **Question***: does there exist a substitution θ that is grounding for the terms such that $t_1\theta \xrightarrow{*}_{R_1} s_1$ and $t_2\theta \xrightarrow{*}_{R_2} s_2\theta$?*

We believe that this result is of independent interest. It provides a partial answer to the open problem regarding decidability of simultaneous rigid E-unification with two rigid equations [11, 24, 28]. We then show that the following restricted cases of second-order unification are undecidable:

- *There are two second-order variables, each occurring twice.*
- *There is one second-order variable that occurs four times.*
- *There is one second-order variable that occurs five times and each occurrence has ground arguments.*

Hence, we obtain a precise bound between the undecidable and the decidable cases of second-order unification where each second-order variable is allowed to occur at most twice, since the problem is decidable with one second-order variable [15]. In Section 6 we review the main results known about second-order unification and mention some open problems and directions for future research.

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 [7] or Goldfarb [10].

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.

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. L is *monadic* if all function symbols in Σ_L have arity ≤ 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. 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 $F(\boldsymbol{t})$, where F is a second-order variable with arity m and $\boldsymbol{t}$ is a sequence of m terms, the elements of $\boldsymbol{t}$ are called the *arguments* of F. A (second-order) term is called *simple* if 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 $s \to t$.[1] An equation or a rule is *ground* (*simple*) if the terms in it are ground (simple). 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

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

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 $l \to r$ 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 [5].

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 [10] and Farmer [7]. 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 $\leq n$. 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 $\boldsymbol{t} = t_1, \ldots, t_n$ of terms, we write $\boldsymbol{t}\theta$ for $t_1\theta, \ldots, t_n\theta$.

Let E be a system of equations in L. The *degree* of E (or L) is the maximum arity of the second-order variables in E (or 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. By SOU *with one second-order variable* we mean the unification problem for second-order languages L such that $|\mathcal{F}_L| = 1$.

2.2 Context Unification and Linear SOU

Linear SOU [14] is SOU where substitutions are required to map second-order variables of arity n to closed terms with exactly one occurrence of each bound

variable z_i for $i \leq n$. *Context Unification*[1, 20, 22] is Linear SOU for languages of degree 1.

2.3 Simultaneous Rigid *E*-Unification

Let L be a first-order language. A *rigid equation* in L is a pair (E, e), where E is a system of equations in L and e is an equation in L. *Simultaneous Rigid E-Unification* [9], or *SREU*, *for* L is the following decision problem:

- **Given:** a system $\{ (E_i, e_i) \mid 1 \leq i \leq n \}$ of rigid equations.
- **Question:** Does there exist a substitution θ that is grounding for each E_i and e_i such that $E_i\theta \models e_i\theta$ for $1 \leq i \leq n$?

By SREU we mean SREU for arbitrary first-order languages.

3 A Relation Between Rewriting and SOU

We use the following lemma to relate certain rewriting problems to second-order unification. The key construction in the lemma is due to Levy. A proof is given in Levy [15] (of a similar statement), and in Veanes [26]. The basic techniques involved in the construction have their roots in Goldfarb [10] and Farmer [7].

Given a set of rules R, with a fixed enumeration $\{ l_i \to r_i \mid 1 \leq i \leq m \}$ of the rules, we write $\boldsymbol{l}_R$ for the sequence $l_1, \ldots, l_m$ and $\boldsymbol{r}_R$ for the sequence $r_1, \ldots, r_m$.

Lemma 1. *Let s, t, R, c, f, F be as follows:*

- *s and t are terms and R is a system of rules in a language L;*
- *c is a constant and f is a binary function symbol such that $c, f \notin \Sigma_L$;*
- *F is a second-order variable with arity $|R| + 1$ such that $F \notin \mathcal{F}_L$;*

The following statements are equivalent for all θ such that $F \notin \mathrm{dom}(\theta)$, and $s\theta$ and $R\theta$ are ground and in L.

(i) Some extension of θ with F solves $F(\boldsymbol{l}_R, f(s, c)) \approx f(t, F(\boldsymbol{r}_R, c))$.
(ii) $t\theta \xrightarrow{*}_{R\theta} s\theta$.

Note that, if R is ground then the condition that $R\theta$ is a set of ground rules in L is trivially satisfied for all θ. Similarly for s.

4 One Second-Order Variable is Enough

The principal result of this section is that the number of *different* second-order variables in second-order unification plays a minor role compared to the total number of *occurrences* of 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 technique used in Farmer [7].

Several important properties are preserved by that reduction: *degree*, *number of occurrences of second-order variables* and *simplicity*.

An *interpolation equation* is a second-order equation of the form $F(\boldsymbol{t}) \approx s$, where F is a second-order variable, $\boldsymbol{t}$ is a sequence of first-order terms, and s is a first-order term. A system S of second-order equations is said to be in *reduced form* if each equation in S is either first-order or an interpolation equation.

Given a system $\mathcal{S}$ of second-order equations, let $\deg(\mathcal{S})$ denote the degree of $\mathcal{S}$ and $\mathrm{occ}(\mathcal{S})$ the total number of occurrences of second-order variables in $\mathcal{S}$. The following fact is well-known and easy to prove.

Lemma 2. *Let $\mathcal{S}$ be a system of second-order equations. There is a system $\mathcal{S}_{\mathrm{red}}$ in reduced form that is solvable if and only if $\mathcal{S}$ is solvable. Moreover,* $\deg(\mathcal{S}_{\mathrm{red}}) = \deg(\mathcal{S})$, $\mathrm{occ}(\mathcal{S}_{\mathrm{red}}) = \mathrm{occ}(\mathcal{S})$, *and if $\mathcal{S}$ is simple then so is $\mathcal{S}_{\mathrm{red}}$.*

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 degree of the original system. Let $\mathcal{S}$ be a system of interpolation equations:

$$\bigcup_{1\le i\le m}\ \bigcup_{1\le j\le k_i} F_i(\boldsymbol{s}_{ij}) \approx t_{ij},$$

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. (If some F_i has arity $< \deg(\mathcal{S})$ then increase the arity of F_i to $\deg(\mathcal{S})$ and replace each $F_i(\boldsymbol{s}_{ij})$ by $F_i(\boldsymbol{s}_{ij}, s, \ldots, s)$, where s is the last term in the sequence $\boldsymbol{s}_{ij}$. Clearly, this neither affects the solvability of $\mathcal{S}$, nor the other properties we are interested in, like simplicity.)

Let G be a second-order variable with arity $\deg(S)$, let g be a new m-ary function symbol, and let S_{one} denote the system of second-order equations consisting of

$$\bigcup_{1\le i\le m}\ \bigcup_{1\le j\le k_i} G(\boldsymbol{s_{ij}}) \approx g(\underbrace{\_, \ldots, \_}_{i-1}, t_{ij}, \underbrace{\_, \ldots, \_}_{m-i})$$

and, unless all elements of some sequence $\boldsymbol{s}_{ij}$ are nonvariables[2], the equation

$$G(c, c, \ldots, c) \approx g(\_, \_, \ldots, \_),$$

where c is a constant and each occurrence of '$\_$' denotes a new first-order variable. By combining Lemma 2 with the latter reduction we can prove the following result.

Theorem 3. *Let S be a system of second-order equations. There is a reduced system S_{one}, with at most one second-order variable, such that S_{one} is solvable if and only if S is solvable. Moreover,* $\deg(S_{\mathrm{one}}) = \deg(S)$, $\mathrm{occ}(S) \le \mathrm{occ}(S_{\mathrm{one}}) \le \mathrm{occ}(S) + 1$, *and if S is simple then so is S_{one}.*

[2] This condition can be considerably weakened.

Proof. Assume, without loss of generality, that S consists solely of interpolation equations. Clearly, the additional conditions are satisfied. We prove that S is solvable if and only if S_{one} is solvable. We just consider a special case. The proof of the general case is analogous. Let S be the following system:

$$\begin{aligned} F_1(\boldsymbol{s}_1) &\approx t_1 \\ F_2(\boldsymbol{s}_2) &\approx t_2 \\ F_3(\boldsymbol{s}_3) &\approx t_3 \end{aligned}$$

Then S_{one} is the following system:

$$\begin{aligned} G(\boldsymbol{s}_1) &\approx g(t_1, x_{12}, x_{13}) \\ G(\boldsymbol{s}_2) &\approx g(x_{21}, t_2, x_{23}) \\ G(\boldsymbol{s}_3) &\approx g(x_{31}, x_{32}, t_3) \\ (\quad G(\boldsymbol{c}) &\approx g(x_1, x_2, x_3) \quad) \end{aligned}$$

(⇒) Assume θ solves S. Define θ' as follows. For all first-order variables in S, θ' agrees with θ. For G, $G\theta' = g(F_1\theta, F_2\theta, F_3\theta)$. For the new first-order variables $\{x_{12}, x_{13}, x_{21}, x_{23}, x_{31}, x_{32}\}$, let $\theta'(x_{ij}) = F_j\theta[\boldsymbol{s}_i\theta]$. For $\{x_1, x_2, x_3\}$, let $\theta'(x_j) = F_j\theta[\boldsymbol{c}]$. Consider the first equation of S_{one}. We have that

$$\begin{aligned} G(\boldsymbol{s}_1)\theta' &= G\theta'[\boldsymbol{s}_1\theta'] \\ &= g(F_1\theta, F_2\theta, F_3\theta)[\boldsymbol{s}_1\theta] \\ &= g(F_1\theta[\boldsymbol{s}_1\theta], F_2\theta[\boldsymbol{s}_1\theta], F_3\theta[\boldsymbol{s}_1\theta]) \\ &= g(t_1\theta, x_{12}\theta', x_{13}\theta') \\ &= g(t_1, x_{12}, x_{13})\theta'. \end{aligned}$$

Hence θ' solves $G(\boldsymbol{s}_1) \approx g(t_1, x_{12}, x_{13})$. The other cases are similar.

(⇐) Assume that θ' solves S_{one}, we construct a substitution θ that solves S. So $G\theta' = g(s'_1, s'_2, s'_3)$ for some closed terms s'_1, s'_2 and s'_3 of rank $\leq \deg(S)$, or else θ' wouldn't solve either an equation in S_{one} where all arguments of G are nonvariables (note that g does not occur in those arguments), or the last equation $G(\boldsymbol{c}) \approx g(x_1, x_2, x_3)$.

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

$$G\theta'[\boldsymbol{s}_1\theta'] = g(s'_1[\boldsymbol{s}_1\theta'], s'_2[\boldsymbol{s}_1\theta'], s'_3[\boldsymbol{s}_1\theta']) = g(F_1\theta[\boldsymbol{s}_1\theta], s'_2[\boldsymbol{s}_1\theta], s'_3[\boldsymbol{s}_1\theta]),$$

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

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

But θ' solves S_{one}, and thus $F_1\theta[\boldsymbol{s}_1\theta] = t_1\theta$. Hence θ solves $F_1(\boldsymbol{s}_1) \approx t_1$. The other cases are similar. ⊠

The above reduction implies, by using Goldfarb's result [10], the undecidability of SOU with one second-order variable. Moreover, by using Farmer's main result [7] (that implies the undecidability of SOU already if all second-order variables are unary), the following corollary is immediate.

Corollary 4. *SOU is undecidable already with one unary second-order variable.*

5 New Undecidable Cases of SOU

In this section we show that SOU is undecidable with two second-order variables each occurring twice. We show also that SOU with one second-order variable is undecidable under the following additional restrictions:

1. The equations are simple and the second-order variable occurs 5 times.
2. The second-order variable occurs 4 times.

The above two results (that are incomparable) should be contrasted with the following subcases of SOU with one second-order variable, that have recently either been proved or claimed to be *decidable*:

1*. The equations are simple and there are no first-order variables. In particular, Comon [2] claims that solvability of simple equations of the form $F(\boldsymbol{s}) \approx t$, where F may occur in t and there are no other variables in the equation, can be proved to be decidable by using finite tree automata techniques.
2*. The case when the second-order variable occurs at most 2 times is proved decidable in Levy [15].

When more than one second-order variables are allowed, then undecidability arises already if there are two second-order variables, one of them occurs twice, the other one occurs three times, and there are no other variables [26] (see Lemma 5 below). In comparison with Statements 1 and 1*, it is interesting to note that the decidability of simultaneous rigid E-unification with one variable was recently settled by using finite tree automata techniques [3].

5.1 Simple Equations

We use the following result from Veanes [26] and apply Theorem 3 to prove Statement 1 above.

Lemma 5. *There is a system $S^{\mathrm{u}}(x, F_1, F_2)$ of simple second-order equations:*

$$\{F_1(\boldsymbol{s}_1) \approx f_1(x, F_1(\boldsymbol{s}_2)), \quad F_2(\boldsymbol{s}_4) \approx f_2(F_1(\boldsymbol{s}_3), F_2(\boldsymbol{s}_5))\}$$

such that the problem of determining whether $S^{\mathrm{u}}(t, F_1, F_2)$ is solvable for a given ground term t, is undecidable.

Observe that the system $S^{\mathrm{u}}(t, F_1, F_2)$ is solvable if and only if the following system of interpolation equations is solvable:

$$\begin{aligned} F_1(\boldsymbol{s}_1) &\approx f_1(t, x_1) \\ F_1(\boldsymbol{s}_2) &\approx x_1 \\ F_1(\boldsymbol{s}_3) &\approx x_2 \\ F_2(\boldsymbol{s}_4) &\approx f_2(x_2, x_3) \\ F_2(\boldsymbol{s}_5) &\approx x_3 \end{aligned}$$

where $\{x_1, x_2, x_3\}$ are new first-order variables.

Theorem 6. *SOU with one second-order variable is undecidable already when restricted to systems $\mathcal{S}$ of simple equations such that* $\mathrm{occ}(\mathcal{S}) \leq 5$.

Proof. By Theorem 3 and Lemma 5. ⊠

5.2 2 Occurrences of 2 Second-Order Variables

We first prove the undecidability of a decision problem that is closely related to simultaneous rigid E-unification with ground left-hand sides and two rigid equations (Theorem 7). We then apply Lemma 1 and Theorem 3. The proof of Theorem 7 uses a powerful technique that is used in a similar context in Plaisted [19], Gurevich and Veanes [11] and Veanes [24–26].

Theorem 7. *There are two ground rewrite systems R_1 and R_2 such that R_1 is canonical, R_2 is noetherian, and the following decision problem is undecidable.*

- **Given***: first-order terms s_1, t_1, s_2, t_2, where s_1 is ground and all variables in s_2 or t_2 occur in t_1.*
- **Question***: does there exist a substitution θ that is grounding for the terms such that $t_1\theta \xrightarrow{*}_{R_1} s_1$ and $t_2\theta \xrightarrow{*}_{R_2} s_2\theta$?*

Proof. The main idea is as follows. We consider a fixed universal Turing machine M and construct the systems R_1 and R_2 with the following properties.

1. The system R_1 is such that, roughly, given appropriate terms t_1 and s, where y is a variable in t_1 and s is ground, $t_1\theta \xrightarrow{*}_{R_1} s$ if and only if $y\theta$ represents a sequence of moves of M:

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

   i.e., v_i^+ is the successor of v_i according to the transition function of M.
2. The system R_2 is such that, roughly, given appropriate terms t_2 and s_2 (including the variable y), $t_2\theta \xrightarrow{*}_{R_2} s_2\theta$ if and only if $y\theta$ is a shifted pairing of its first projection. (See Figure 1.)

Both cases together imply that $y\theta$ represents a valid sequence of moves of M if and only if M accepts the given input string (encoded in the terms). ⊠

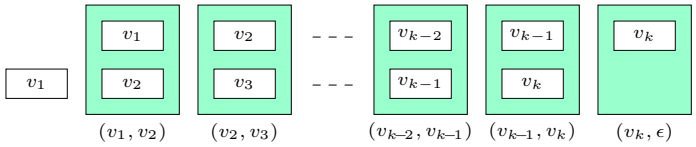

Fig. 1. $((v_1, v_2), (v_2, v_3), \ldots, (v_k, \epsilon))$ is a *shifted pairing* of $(v_1, v_2, \ldots, v_k)$.

Theorem 7 provides a partial answer to the open problem of whether SREU *with two rigid equations* is decidable or not [11, 24, 28]. The problem stated in Theorem 7 with the additional condition that R_2 is *confluent* (and thus canonical), corresponds to a very strong subcase of this problem. We now turn to the main result of this section.

Theorem 8. *SOU is undecidable with 2 second-order variables each occurring 2 times.*

Proof. Let R_1 and R_2 be the rewrite systems in Theorem 7, in language L say. Let s_1, t_1, s_2, t_2 be first-order terms in L, such that s_1 is ground and all variables in the terms occur in t_1. Let f be a new binary function symbol and c a new constant, and let F_i be a second-order variable with arity $|R_i| + 1$ for $i = 1, 2$. Let S be the following system of second-order equations:

$$F_i(\boldsymbol{l}_{R_i}, f(s_i, c)) \approx f(t_i, F_i(\boldsymbol{r}_{R_i}, c)) \quad i = 1, 2.$$

We prove that S is solvable if and only if there exists a grounding substitution θ such that $t_i\theta \xrightarrow{*}_{R_i} s_i\theta$ for $i = 1, 2$, and apply Theorem 7 to complete this proof.

(⇐) Assume there exists a substitution θ such that $t_i\theta \xrightarrow{*}_{R_i} s_i\theta$ for $i = 1, 2$. Then $t_1\theta \xrightarrow{*}_{R_1} s_1$, and thus $t_1\theta \in \mathcal{T}_{\Sigma_L}$. Hence $s_2\theta \in \mathcal{T}_{\Sigma_L}$, since all variables in s_2 occur in t_1. Now apply Lemma 1.

(⇒) Assume that $\theta \cup \{F_1 \mapsto t_1', F_2 \mapsto t_2'\}$ solves S. Since R_1 and s_1 are ground, by Lemma 1, $t_1\theta \xrightarrow{*}_{R_1} s_1$. This implies that $s_2\theta \in \mathcal{T}_{\Sigma_L}$, and thus, again by Lemma 1, $t_2\theta \xrightarrow{*}_{R_2} s_2\theta$. ⊠

By using Theorem 3 we get the following corollary of Theorem 8.

Corollary 9. *SOU is undecidable with 1 second-order variable that occurs 4 times.*

6 Current Status of SOU and Open Problems

In the following we give a (roughly) chronological list of the main results known about SOU, including the new results proved in this paper. At the end of the section we mention some open problems. The undecidability of higher-order unification in general, in fact third-order unification, is proved (independently) in Huet [12] and in Lucchesi [16].

1. In 1981 Goldfarb shows that SOU is undecidable [10], by reduction from Hilbert's tenth problem.
2. In 1988 Farmer shows that Monadic SOU is decidable [6], by reduction to word equations (Makanin 1977 [17]).
3. In 1991 Farmer shows that there is an integer n such that the undecidability of SOU holds for all nonmonadic languages with more than n second-order variables. In particular, SOU is undecidable even if all second-order variables are unary.
4. Some (incomparable) cases of context unification are proved to be decidable in Comon [1] and Schmidt-Schauß [20]. Recent developments towards a more general decidability result are discussed in Schmidt-Schauß and Schulz [22].
5. Some cases of linear SOU are proved to be decidable in Levy [14], in particular when each variable occurs at most twice.
6. Schubert claims that SOU is undecidable for systems of simple equations and provides a very complicated argument to support this claim [23].
7. A natural reduction of SOU to SREU is given in Degtyarev and Voronkov [4]. Converse reductions are given in Levy [15] and in Veanes [26], implying that SOU and SREU are in fact polynomial time (even logspace) equivalent.
8. SOU where each second-order variable occurs at most twice is proved undecidable in Levy [15]. Here we have improved this result, implying the following clear boundary:
   - The case is undecidable with 2 second-order variables (Theorem 8).
   - The case is decidable with 1 second-order variable [15].
9. Veanes proves that there is an integer n such that SOU is undecidable for all nonmonadic languages with at least two second-order variables with arities $\geq n$, already for systems of simple equations [26].
10. SOU with one second-order variable is decidable if the second-order variable occurs ≤ 2 times [15], and undecidable if the second-order variable occurs 4 times (Corollary 9). Also, SOU with one unary second-order variable is undecidable (Corollary 4).
11. SOU with one second-order variable and simple equations, is claimed to be decidable if first-order variables are disallowed [2], otherwise it is undecidable already if the second-order variable occurs 5 times (Theorem 6).

The above list of results is not exhaustive. Due to the strong connection between SOU 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 SOU. The most recent survey discussing such relations is given in Voronkov [28].

Currently, we are working on a decidability proof of SOU where each (first or second-order) variable occurs at most twice. This case is polynomial time equivalent to SREU with exactly the same restriction. We note that this case is at least as hard as (nonsimultaneous) rigid E-unification, and thus NP-hard [8], because there is a straightforward reduction from an arbitrary rigid equation to an equivalent system of two rigid equations where each variable occurs at most twice.

The decidability of context unification, or the more general linear SOU, is a difficult open problem. In all undecidability proofs of the restricted cases of SOU that we have considered, a common key feature is the unboundedness of the number of occurrences of bound variables in solutions (in particular, nonlinearity). It might be worthwhile to study the relation between SOU and SREU more carefully and to identify the restrictions on SREU corresponding to linearity.

References

1. H. Comon. Completion of rewrite systems with membership constraints Part I: Deduction rules. Preliminary version of a paper to appear in *J. Symbolic Computation*, 1997.
2. H. Comon, March 1998. Personal communication.
3. A. Degtyarev, Yu. Gurevich, P. Narendran, M. Veanes, and A. Voronkov. The decidability of simultaneous rigid E-unification with one variable. In T. Nipkow, editor, *Rewriting Techniques and Applications*, volume 1379 of *Lecture Notes in Computer Science*, pages 181–195. Springer Verlag, 1998.
4. A. Degtyarev and A. Voronkov. The undecidability of simultaneous rigid E-unification. *Theoretical Computer Science*, 166(1–2):291–300, 1996.
5. 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.
6. W.M. Farmer. A unification algorithm for second-order monadic terms. *Annals of Pure and Applied Logic*, 39:131–174, 1988.
7. W.M. Farmer. Simple second-order languages for which unification is undecidable. *Theoretical Computer Science*, 87:25–41, 1991.
8. 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.
9. 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.
10. W.D. Goldfarb. The undecidability of the second-order unification problem. *Theoretical Computer Science*, 13:225–230, 1981.
11. 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*.
12. G. Huet. The undecidability of unification in third order logic. *Information and Control*, 22:257–267, 1973.
13. J. Levy and J. Agustí. Bi-rewrite systems. *J. of Symbolic Computation*, 22:279–314, 1996.
14. 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.
15. J. Levy. Decidable and undecidable second-order unification problems. In T. Nipkow, editor, *Rewriting Techniques and Applications*, volume 1379 of *Lecture Notes in Computer Science*, pages 47–60. Springer Verlag, 1998.
16. C.L. Lucchesi. The undecidability of the unification problem for third order languages. Report CSRR 2059, Department of Applied Analysis and Computer Science, University of Waterloo, 1972.

17. 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.
18. D. Miller. Unification of simply typed lambda-terms as logic programming. In K. Furukawa, editor, *Proceedings of the Eighth International Conference on Logic Programming*, pages 255–269, Paris, France, June 1991. MIT Press.
19. 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.
20. M. Schmidt-Schauß. Unification of stratified second-order terms. Technical Report 12/94, Johan Wolfgang-Göthe-Universität, Frankfurt, 1995.
21. M. Schmidt-Schauß. An algorithm for distributive unification. In *Rewriting Techniques and Applications*, volume 1103 of *Lecture Notes in Computer Science*, pages 287–301. Springer Verlag, 1996.
22. M. Schmidt-Schauß and Klaus U. Schulz. On the exponent of periodicity of minimal solutions of context equations. In T. Nipkow, editor, *Rewriting Techniques and Applications*, volume 1379 of *Lecture Notes in Computer Science*, pages 61–75. Springer Verlag, 1998.
23. A. Schubert. 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*, pages 279–288, San Diego, California, January 1998. ACM Press.
24. M. Veanes. *On Simultaneous Rigid E-Unification*. PhD thesis, Computing Science Department, Uppsala University, 1997.
25. 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.
26. M. Veanes. The relation between second-order unification and simultaneous rigid E-unification. Research Report MPI-I-98-2-005, Max-Planck-Institut für Informatik, Im Stadtwald, D-66123 Saarbrücken, Germany, February 1998. The short version of this paper appears in *Proc. LICS'98*.
27. 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.
28. 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*.

A Proof of Theorem 7

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. 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., $q_\mathrm{f}^+ = \epsilon$).

A.1 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 $v = a_1a_2\cdots a_m$ be any ID of M and let $v^+ = b_1b_2\cdots b_n$. (Note that $m-1 \le n \le m+1$.) We let $\langle v, v^+\rangle$ denote the following string:

$$\langle v, v^+\rangle = \begin{cases} c_{a_1b_1}c_{a_2b_2}\cdots c_{a_mb_m}c_{\flat b_n}, & \text{if } n = m+1;\\ c_{a_1b_1}c_{a_2b_2}\cdots c_{a_nb_n}c_{a_m\flat}, & \text{if } n = m-1;\\ c_{a_1b_1}c_{a_2b_2}\cdots c_{a_mb_n}, & \text{if } n = m. \end{cases}$$

we call such a string a *move* also. (Note that $\langle q_\mathrm{f}, \epsilon\rangle = c_{q_\mathrm{f}\flat}$.) 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:

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

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:

$$f_\mathrm{w}(a_1, f_\mathrm{w}(a_2, \ldots, f_\mathrm{w}(a_n, c_\mathrm{w})\cdots)) = \text{“}a_1a_2\cdots a_n\text{”},$$

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 follows from [11, 24, Train Theorem]. A signature Σ' is a *constant-expansion* of a signature Σ if $\Sigma \subseteq \Sigma'$ and $\Sigma' \setminus \Sigma$ is a set of constants.

Lemma 10. *There is a canonical system R_{mv} of ground rules over a constant-expansion of Σ_{mv}, such that for all ground terms t over Σ_{mv}, t is a move-train if and only if $t \xrightarrow{*}_{R_{\mathrm{mv}}} c_{\mathrm{t}}$.*

A.2 First component: move-trains

For terms $t_1, \ldots, t_k$ we write $\langle t_1, \ldots, t_k \rangle$ for a tuple. For any signature Σ that we consider below, ground rewrite system R over Σ, and terms $s_i, t_i \in \mathcal{T}_\Sigma$ for $1 \leq i \leq k$, it is assumed that

$$\langle t_1, \ldots, t_k \rangle \xrightarrow{*}_R \langle s_1, \ldots, s_k \rangle \quad \Leftrightarrow \quad \bigwedge_{1 \leq i \leq k} t_i \xrightarrow{*}_R s_i.$$

We make use of the following basic property of ground rewrite systems. Two signatures or sets of rules are *constant-disjoint* if they do not share any constants.

Lemma 11. *Let R_i, for $1 \leq i \leq k$, be ground rewrite systems over corresponding constant-disjoint signatures Σ_i, for $1 \leq i \leq k$. Let s be a ground term over some Σ_j, where $1 \leq j \leq k$, and t a ground term. Then*

$$t \xrightarrow{*}_{\bigcup_{1 \leq i \leq k} R_i} s \quad \Leftrightarrow \quad t \xrightarrow{*}_{R_j} s.$$

By using Lemma 10, let R_{mv} be a set of ground rules over a constant-expansion Σ'_{mv} of Σ_{mv} such that for all $t \in \mathcal{T}_{\Sigma_{\mathrm{mv}}}$,

$$t \xrightarrow{*}_{R_{\mathrm{mv}}} c_{\mathrm{t}} \quad \Leftrightarrow \quad t \text{ is a move-train.}$$

For a natural number l and a signature Σ we write $\Sigma^{(l)}$ for the constant-disjoint copy of Σ where each constant c has been replaced with a new constant $c^{(l)}$, we say that $c^{(l)}$ *has label* l. For $t \in \mathcal{T}_\Sigma$ and a set of ground rules R over Σ, we define $t^{(l)} \in \mathcal{T}_{\Sigma^{(l)}}$ and $R^{(l)}$ over $\Sigma^{(l)}$ accordingly.

Let R_1 be the following system of ground rewrite rules:

$$R_1 = \mathrm{Gr}(\Sigma'_{\mathrm{id}}) \cup R^{(0)}_{\mathrm{mv}} \cup R^{(1)}_{\mathrm{mv}} \cup R^{(2)}_{\mathrm{mv}}.$$

The following lemma is an easy corollary of the above definitions and Lemma 11.

Lemma 12. *For all ground terms s, t_0, t_1, and t_2,*

$$\langle s, t_0, t_1, t_2 \rangle \xrightarrow{*}_{R_1} \langle c_{\mathrm{t}}, c^{(0)}_{\mathrm{t}}, c^{(1)}_{\mathrm{t}}, c^{(2)}_{\mathrm{t}} \rangle$$

if and only if $s \in \mathcal{T}_{\Sigma'_{\mathrm{id}}}$ and $t_l \xrightarrow{*}_{R^{(l)}_{\mathrm{mv}}} c^{(l)}_{\mathrm{t}}$ for $l \in \{0, 1, 2\}$.

A.3 Second component: shifted pairing

Let A_1, and A_2 be the following sets of ground rules:

$$A_1 = \{\, c^{(1)} \to c^{(0)} \mid c \text{ a constant in } \Sigma_{\mathrm{mv}} \,\}$$
$$A_2 = \{\, c^{(2)} \to c^{(0)} \mid c \text{ a constant in } \Sigma_{\mathrm{mv}} \,\}$$

The following lemma is easy to prove.

Lemma 13. *For all $t_0, t_1, t_2 \in \Sigma'_{\mathrm{mv}}$,*

$$t_1^{(1)} \xrightarrow{*}_{A_1} t_0^{(0)}, \quad t_2^{(2)} \xrightarrow{*}_{A_2} t_0^{(0)} \quad \Leftrightarrow \quad t_0 = t_1 = t_2 \in \mathcal{T}_{\Sigma_{\mathrm{mv}}}.$$

Given a move-train t that represents a nonempty move sequence, say

$$t = \text{“}(\langle v_1, v_1^+\rangle, \langle v_2, v_2^+\rangle, \ldots, \langle v_k, v_k^+\rangle)\text{”},$$

define the *first projection* of t as the following train

$$\pi_1(t) = \text{“}(v_1, v_2, \ldots, v_k)\text{”},$$

and the *second projection* of t as the following train

$$\pi_2(t) = \begin{cases} \text{“}(v_1^+, v_2^+, \ldots, v_{k-1}^+)\text{”}, \text{ if } v_k^+ = \epsilon; \\ \text{“}(v_1^+, v_2^+, \ldots, v_k^+)\text{”}, \quad \text{otherwise.} \end{cases}$$

We say that t is the *shifted pairing* of its first projection if $\pi_1(t) = f_{\mathrm{t}}(\text{“}v_1\text{”}, \pi_2(t))$ and we refer to v_1 as the *first ID of t*. The next lemma follows from the fact that if a move-train is a shifted pairing of its first projection then the first projection represents a valid computation of M. Recall that q_0 is the initial state of M.

Lemma 14. *Let v be an input string for M. Then M accepts v if and only if there exists a move-train t with first ID $q_0 v$ such that t is the shifted pairing of its first projection.*

The following system of ground rules is used for obtaining the first projection of a move-train.

$$\begin{aligned} \Pi_1 = {} & \{\, c_{ab}^{(1)} \to a \mid a, b \in \Sigma_M, a \neq \text{ƀ} \,\} \cup \\ & \{\, \text{“}c_{\text{ƀ}b}\text{”}^{(1)} \to c_{\mathrm{w}} \mid b \in \Sigma_M \,\} \cup \\ & \{c_{\mathrm{w}}^{(1)} \to c_{\mathrm{w}}, c_{\mathrm{t}}^{(1)} \to c_{\mathrm{t}}\} \end{aligned}$$

The correctness of the following lemma is easily seen from the definitions.

Lemma 15. *For all move-trains t and $s \in \mathcal{T}_{\Sigma'_{\mathrm{id}}}$,*

$$t^{(1)} \xrightarrow{*}_{\Pi_1} s \quad \Leftrightarrow \quad s = \pi_1(t).$$

The following rule system is used for obtaining the second projection.

$$\begin{aligned}\Pi_2 = &\{\, c^{(2)}_{ab} \to b \mid a, b \in \Sigma_M, b \neq \flat \,\} \cup \\ &\{\, \text{“}c_{a\flat}\text{”}^{(2)} \to c_{\mathrm{w}} \mid a \in \Sigma_M \,\} \cup \\ &\{f_{\mathrm{t}}(\text{“}\langle q_{\mathrm{f}}, \epsilon\rangle\text{”}^{(2)}, c^{(2)}_{\mathrm{t}}) \to c_{\mathrm{t}}\} \cup \\ &\{c^{(2)}_{\mathrm{w}} \to c_{\mathrm{w}}, c^{(2)}_{\mathrm{t}} \to c_{\mathrm{t}}\}\end{aligned}$$

Lemma 16. *For all move-trains t and IDs v,*

$$f_{\mathrm{t}}(\text{“}v\text{”}, t^{(2)}) \xrightarrow{*}_{\Pi_2} \pi_1(t) \quad \Leftrightarrow \quad \pi_1(t) = f_{\mathrm{t}}(\text{“}v\text{”}, \pi_2(t)).$$

Proof. Let $t = \text{“}(\langle v_1, v_1^+\rangle, \ldots, \langle v_k, v_k^+\rangle)\text{”}$, $s = \pi_1(t)$, and v an ID.

(⇒) Assume that $f_{\mathrm{t}}(\text{“}v\text{”}, t^{(2)}) \xrightarrow{*}_{\Pi_2} s$. Thus $t^{(2)} \xrightarrow{*}_{\Pi_2} \text{“}(v_2, \ldots, v_k)\text{”}$ and $v = v_1$. So $\langle v_k, v_k^+\rangle$ must be $\langle q_{\mathrm{f}}, \epsilon\rangle$ because the only way to reduce the length of $t^{(2)}$ in Π_2 is by using the rule $f_{\mathrm{t}}(\text{“}\langle q_{\mathrm{f}}, \epsilon\rangle\text{”}, c_{\mathrm{t}})^{(2)} \to c_{\mathrm{t}}$. Moreover, it follows that $\text{“}\langle v_i, v_i^+\rangle\text{”}^{(2)} \xrightarrow{*}_{\Pi_2} \text{“}v_{i+1}\text{”}$, and thus $v_i^+ = v_{i+1}$ for $1 \leq i < k$.

(⇐) Assume $\pi_1(t) = f_{\mathrm{t}}(\text{“}v\text{”}, \pi_2(t))$. Then $v_k^+ = \epsilon$ and hence $v_k = q_{\mathrm{f}}$, and thus $f_{\mathrm{t}}(\text{“}\langle v_k, v_k^+\rangle\text{”}, c_{\mathrm{t}})^{(2)} \longrightarrow_{\Pi_2} c_{\mathrm{t}}$. Moreover, $v_i^+ = v_{i+1}$ for $1 \leq i < k$. The rest is obvious from the fact that $\text{“}\langle v_i, v_i^+\rangle\text{”}^{(2)} \xrightarrow{*}_{\Pi_2} \text{“}v_i^+\text{”}$ for $1 \leq i < k$. ⊠

We use the following lemma for shifted pairing.

Lemma 17. *Let t be a move-train, $s \in \mathcal{T}_{\Sigma'_{\mathrm{id}}}$, and v an ID. Then $t^{(1)} \xrightarrow{*}_{\Pi_1} s$ and $f_{\mathrm{t}}(\text{“}v\text{”}, t^{(2)}) \xrightarrow{*}_{\Pi_2} s$ if and only if t is the shifted pairing of s with first ID v.*

Proof. By Lemma 15 and Lemma 16, $s = \pi_1(t) = f_{\mathrm{t}}(\text{“}v\text{”}, \pi_2(t))$. ⊠

We now combine the above rule systems into one system

$$R_2 = A_1 \cup A_2 \cup \Pi_1 \cup \Pi_2$$

and use the following lemma to extract the respective subsystems from R_2.

Lemma 18. *For all $s \in \mathcal{T}_{\Sigma'_{\mathrm{id}}}$, $t_l \in \mathcal{T}_{\Sigma'^{(l)}_{\mathrm{mv}}}$ for $l \in \{0, 1, 2\}$, and IDs v,*

$$\langle t_1, t_2, t_1, f_{\mathrm{t}}(\text{“}v\text{”}, t_2)\rangle \quad \xrightarrow{*}_{R_2} \quad \langle t_0, t_0, s, s\rangle$$

if and only if

$$\begin{aligned} t_l &\xrightarrow{*}_{A_l} t_0, \quad (l \in \{1, 2\}) \\ t_1 &\xrightarrow{*}_{\Pi_1} s, \\ f_{\mathrm{t}}(\text{“}v\text{”}, t_2) &\xrightarrow{*}_{\Pi_2} s. \end{aligned}$$

Proof. The only nontrivial direction is (⇒). Consider $t_1 \xrightarrow{*}_{R_2} t_0$. This reduction cannot use $A_2 \cup \Pi_2$ since t_1 doesn't include constants with label 2 and no rule in R_2 has a constant with label 2 on the right-hand side. Furthermore, this reduction cannot use Π_1, or else a constant from Σ_M appears in t_0, contradicting that all constants in t_0 have label 0. Hence $t_1 \xrightarrow{*}_{A_1} t_0$. An analogous statement shows the other cases. ⊠

A.4 Combined construction

The following is the main lemma.

Lemma 19. *For any input string v for M, M accepts v if and only if there exists a substitution θ grounding for x, y_0, y_1, and y_2 such that*

$$\langle x, y_0, y_1, y_2\rangle\theta \xrightarrow{*}_{R_1} \langle c_{\mathrm{t}}, c_{\mathrm{t}}^{(0)}, c_{\mathrm{t}}^{(1)}, c_{\mathrm{t}}^{(2)}\rangle,$$
$$\langle y_1, y_2, y_1, f_{\mathrm{t}}(\text{“}q_0 v\text{”}, y_2)\rangle\theta \xrightarrow{*}_{R_2} \langle y_0, y_0, x, x\rangle\theta.$$

Proof. Let v be given. By Lemma 12 and Lemma 18 it is enough to prove that M accepts v if and only if there exists a θ such that $x\theta \in \mathcal{T}_{\Sigma'_{\mathrm{id}}}$ and

$$y_l\theta \xrightarrow{*}_{R^{(l)}_{\mathrm{mv}}} c_{\mathrm{t}}^{(l)}, \quad (l \in \{0,1,2\}), \tag{1}$$
$$y_l\theta \xrightarrow{*}_{A_l} y_0\theta, \quad (l \in \{1,2\}), \tag{2}$$
$$y_1\theta \xrightarrow{*}_{\Pi_1} x\theta, \tag{3}$$
$$f_{\mathrm{t}}(\text{“}q_0 v\text{”}, y_2\theta) \xrightarrow{*}_{\Pi_2} x\theta. \tag{4}$$

Note that (1) implies that $y_l\theta$ is a ground term over $\Sigma^{\prime(l)}_{\mathrm{mv}}$ for $l \in \{0,1,2\}$.

(⇐) Let θ satisfying (1–4) be given. It follows from (2) and Lemma 13 that there is a term $t \in \mathcal{T}_{\Sigma_{\mathrm{mv}}}$ such that $y_l\theta = t^{(l)}$ for $l \in \{0,1,2\}$. Hence, (1) implies that t is a move-train. Now (3–4) and Lemma 17 imply that t is a shifted pairing of $x\theta$ with first ID $q_0 v$, and thus M accepts v by Lemma 14.

(⇒) Assume that there is a valid computation of M with initial ID v. Let s represent it and let t be the shifted pairing of s. Let $\theta = \{x \mapsto s, y_0 \mapsto t^{(0)}, y_1 \mapsto t^{(1)}, y_2 \mapsto t^{(2)}\}$. Then (1) holds since t is a move-train, (2) follows from Lemma 13, and (3–4) follow from Lemma 17. ⊠

Theorem 7 follows from Lemma 19 by noting that R_1 is indeed canonical and R_2 noetherian, and letting M be a universal Turing machine.