Decidable and Undecidable Second-Order Unification Problems*

Jordi Levy

Institut d'Investigació en Intel·ligència Artificial
Consejo Superior de Investigaciones Científicas
http://www.iiia.csic.es/~levy

Abstract. There is a close relationship between word unification and second-order unification. This similarity has been exploited for instance for proving decidability of monadic second-order unification. Word unification can be easily decided by transformation rules (similar to the ones applied in higher-order unification procedures) when variables are restricted to occur at most twice. Hence a well-known open question was the decidability of second-order unification under this same restriction. Here we answer this question negatively by reducing simultaneous rigid E-unification to second-order unification. This reduction, together with an inverse reduction found by Degtyarev and Voronkov, states an equivalence relationship between both unification problems.
Our reduction is in some sense reversible, providing decidability results for cases when simultaneous rigid E-unification is decidable. This happens, for example, for one-variable problems where the variable occurs at most twice (because rigid E-unification is decidable for just one equation). We also prove decidability when no variable occurs more than once, hence significantly narrowing the gap between decidable and undecidable second-order unification problems with variable occurrence restrictions.

1 Introduction

Word unification [Mak77, Sch91], linear second-order unification [Lev96], context unification [Com93, SS95] and second-order unification [Pie73] are closely related problems. The relationship between word unification and linear second-order unification becomes clear when we codify a word unification problem, like $F \cdot a \cdot G \stackrel{?}{=} G \cdot a \cdot F$, as a linear second-order unification problem $\lambda x.F(a(G(x))) \stackrel{?}{=} \lambda x.G(a(F(x)))$. The relationship between word unification and second-order unification is not so clear, but was used, for instance, to prove decidability of monadic second-order unification [Far88]. Despite their similarities, word unification is decidable [Mak77], second-order unification is undecidable [Gol81], and the question is open for linear second-order unification and context unification (although it is conjectured to be decidable).

* This work was partially supported by the project MODEL (TIC97-0579-C02-01) funded by the CICYT, and the ESPRIT Basic Research Actions CCL and CONSOLE

T. Nipkow (Ed.): Rewriting Techniques and Applications, RTA-98, LNCS 1379, pp. 47–60, 1998.

Decidability of word unification was an open question for a long time and its proof [Mak77] involves a lot of technicalities. However, it is very easy to prove that it is decidable when no variable occurs more than twice in a problem. The same main ideas were used to prove that linear second-order unification and context unification are decidable when no variable occurs more than twice [Lev96]. Thus, the arising question is, are these ideas applicable to second-order unification? The answer is no. We prove this undecidability result by reduction of another undecidable unification problem: simultaneous rigid E-unification [GRS87, DV96]. This reduction, together with a inverse reduction found by Degtyarev and Voronkov [DV95], states a close relationship between both unification problems. More precisely, proves that both problems are polynomial-time equivalent. Based on the preliminary version of this paper, Veanes [Vea98] also proves some of the results we prove. Other related results and ideas appeared in [Sch97].

Our reduction is in some sense reversible, providing decidability results for cases when simultaneous rigid E-unification is decidable. This happens, for example, for one (second-order) variable problems where the variable occurs at most twice, since (non-simultaneous) rigid E-unification is decidable [GNPS88].

This paper proceeds as follows. In section 2 we introduce all the unification problems we will deal with and some preliminary definitions and notation. In section 3 we prove undecidability of second-order unification, when variables are restricted to occur at most twice, by reduction from simultaneous rigid E-unification. We started our research trying to prove the decidability of the problem. The difficulties we found to achieve this purpose suggested us how we could, in fact, prove undecidability, and which undecidable problem we had to chose. However, since simultaneous rigid E-unification is decidable for one equation, we prove in section 4 decidability for one second-order variable problems. Additionally, in section 4, we also prove decidability for problems where variables occur at most once. This closes the gap between decidable and undecidable second-order unification problems w.r.t. variable occurrence restrictions.

2 Preliminary Definitions

We assume that the reader is familiar with unification problems, second-order typed λ-calculus and related topics. Variables are denoted by capital letters ($X, Y, Z \ldots$ when they are first-order, and $F, G \ldots$ when they are second-order variables), constants are denoted by lower case letters ($a, b \ldots$ when they are 0-ary constants, and $f, g \ldots$ for functions), terms by $t, u, v, w \ldots$ and substitutions by Greek letters $\sigma, \rho, \theta \ldots$. Substitutions are represented by finite sets of variable-term pairs, like $\sigma = [X_1 \mapsto t_1] \cdots [X_n \mapsto t_n]$. The application of a substitution σ to a term t is represented by $\sigma(t)$. Notation $t|_p$ represents subterm at position p of t, and $t[u]_p$ represents term t where subterm at position p has been replaced by u. We assume that any term is second-order typed and is written in $\beta\eta$-long normal form, i.e. any term has the form $\lambda\overline{x}\,.\,a(t_1, \ldots, t_n)$ where $\overline{x}$ is a (possibly empty) list of first-order bound variables, a may be a (at most) third-order

constant, a second-order free variable or a first-order bound variable (in this later case $n = 0$), and t_i are also second-order terms in normal form.

2.1 Word Unification

It is easy to describe a complete[1] (non-terminating) procedure for word unification in terms of *transformation rules* [GS90]. Any state of the process is represented by a pair $\langle S, \sigma \rangle$, where S is the problem and σ the substitution computed until that moment. We proceed by applying a substitution ρ, that transforms the pair into a new one $\langle \rho(S), \rho \circ \sigma \rangle$ where $\rho(S)$ can be later simplified. At some point, more than a rule can be applicable, thus the procedure is not deterministic. We distinguish two kinds of words, *rigid* words (when they start by a constant, like $a \cdot w$) and *flexible* words (when they start by a variable, like $X \cdot w$). Therefore, we have three kinds of equations. For each kind of equation, the set of applicable transformations are as follows:

Rigid-rigid equations, like $a \cdot w_1 \stackrel{?}{=} a \cdot w_2$. Only **simplification rule** is applicable

$$\langle \{a \cdot w_1 \stackrel{?}{=} a \cdot w_2\} \cup S, \sigma \rangle \Rightarrow \langle \{w_1 \stackrel{?}{=} w_2\} \cup S, \sigma \rangle$$

Rigid-flexible equations, like $a \cdot w_1 \stackrel{?}{=} X \cdot w_2$. We can apply two different rules:

- **Projection rule** to instantiate the variable on the head by the empty word $\rho = [X \mapsto \epsilon]$.
- **Imitation rule** to instantiate $\rho = [X \mapsto a \cdot X']$. A fresh variable X' is introduced, and the equation is transformed into a rigid-rigid equation $a \cdot \rho(w_1) \stackrel{?}{=} a \cdot X' \cdot \rho(w_2)$ that is later simplified into $\rho(w_1) \stackrel{?}{=} X' \cdot \rho(w_2)$

Flexible-flexible equations, like $X \cdot w_1 \stackrel{?}{=} Y \cdot w_2$.

If $X = Y$, we can simplify the equation by removing both occurrences of the variable to get $w_1 \stackrel{?}{=} w_2$.

Otherwise we can instantiate one of the variables $\rho = [X \mapsto Y \cdot X']$, introducing a new fresh variable X'. The equation is transformed into $Y \cdot X' \cdot \rho(w_1) \stackrel{?}{=} Y \cdot \rho(w_2)$ and simplified into $X' \cdot \rho(w_1) \stackrel{?}{=} \rho(w_2)$.

If no variable occurs more than twice, after instantiating and simplifying equations, no transformation rule increases the size of the problem (in term of number of symbols). Since, there are finitely many unification problems of a given size (up to variable renaming), we can easily prove decidability of the problem [SS95] under this two-occurrences restriction. In fact, although word unification is infinitary[2], we can prove that, under this restriction, there exists a *finite* representation of the (maybe infinite) set of unifiers. For instance, the problem $X \cdot a \stackrel{?}{=} a \cdot X$ has infinitely many most general unifiers $[X \mapsto a \cdot \cdots \cdot a]$, but we can represent all them by a regular expression $[X \mapsto \epsilon] \circ [X \mapsto a \cdot X]^*$.

[1] Notice that this procedure computes all most general unifiers, but not only most general unifiers.

[2] We can have infinitely many most general unifiers for a given unification problem.

2.2 Second-Order Unification

Pietrzykowski [Pie73] was the first to describe a complete second-order unification procedure. The rules that this procedure uses are quite similar to the rules we have described for word unification. We also distinguish between rigid and flexible second-order normal terms. Given a term in normal form $\lambda\overline{x}\,.\,a(t_1,\ldots,t_n)$, if a is a constant or a bound variable, the term is said to be rigid, and flexible if a is a free variable.

A second-order unification problem is a finite set $\{t_1 \stackrel{?}{=} u_1,\ldots,t_n \stackrel{?}{=} u_n\}$ of pairs of second-order terms. For all our purposes, we can assume that our unification problems do not contain third-order constants or λ-bindings, i.e. we can assume that any term is first-order typed and has the form $a(t_1,\ldots,t_n)$, where a is a second-order free variable or second-order constant, and t_i are also first-order terms. Goldfarb [Gol81] proved that second-order unification, even under this restriction, is undecidable.

If we are only interested in deciding if a problem has a solution or not, and not in finding *all* its most general unifiers, we can simplify Pietrzykowski's procedure (notice that all flexible-flexible equations are solvable). These decision procedures for unification problems are called *pre-unification* procedures. Huet [Hue75] was the first to describe a pre-unification procedure for typed λ-calculus. The set of transformation rules for second-order pre-unification can be easily derived from either Huet's higher-order pre-unification procedure or from Pietrzykowski's second-order unification procedure. This set is as follows:

Simplification rule.

$$\langle\{a(t_1,\ldots,t_n) \stackrel{?}{=} a(u_1,\ldots,u_n)\}\cup S,\sigma\rangle \;\Rightarrow\; \langle \bigcup_{i\in[1..n]} \{t_i \stackrel{?}{=} u_i\}\cup S,\sigma\rangle$$

Projection rule. If we have a rigid-flexible equation like $\lambda\overline{x}\,.\,F(t_1,\ldots,t_n) \stackrel{?}{=} \lambda\overline{x}\,.\,g(u_1,\ldots,u_m)$ we can instantiate

$$[F \mapsto \lambda x_1\ldots\lambda x_n\,.\,x_i] \qquad \text{for some } i\in[1..n]$$

Imitation rule. Or, we can instantiate

$$[F \mapsto \lambda x_1\ldots\lambda x_n\,.\,g(F'_1(x_1,\ldots,x_n),\ldots,F'_m(x_1,\ldots,x_n))]$$

Proposition 1. *The procedure based on the previous transformation rules is a sound and complete pre-unification procedure for second-order unification.*

The previous proposition ensures semi-decidability of second-order unification, so when we say "undecidable" or "not decidable" we always mean semi-decidable.

2.3 Simultaneous Rigid E-Unification

Simultaneous rigid E-unification was introduced in [GRS87] in order to extend the tableau method, the method of matings and other proof methods to first-order logic with equality. After some faulty proofs of its decidability, it was proved to be undecidable in [DV96]. The (non-simultaneous) *rigid E-unification problem* can be formulated as follows. Given a finite set of first-order equations $\{t_i \cong u_i \mid i \in [1..n]\}$ and an equation $v \cong w$, decide if there exists a ground[3] substitution θ such that the formula

$$\theta(t_1) = \theta(u_1) \wedge \cdots \wedge \theta(t_n) = \theta(u_n) \Rightarrow \theta(v) = \theta(w)$$

is provable in first-order logic with equality. An instance of the problem is denoted by $t_1 \cong u_1 \wedge \cdots \wedge t_n \cong u_n \vdash_\forall v \cong w$, and is called a *rigid equation*. *Simultaneous rigid E-unification* is formalised as the problem of finding a simultaneous solution for a finite set of rigid equations.

For simplicity, we will also introduce a different notion of rigid unification called **rigid O-unification**. Given a rigid O-unification problem $\bigwedge_{i\in[1..n]} t_i \subseteq u_i \vdash_\forall v \subseteq w$, we say it is solvable if there exists a ground substitution θ such that the formula $\left(\bigwedge_{i\in[1..n]} \theta(t_i) \subseteq \theta(u_i)\right) \Rightarrow \theta(v) \subseteq \theta(w)$ is provable in first-order logic with a monotonic pre-order relation, i.e. without considering the symmetry rule.

Since a rewriting system defines a monotonic pre-order relation, we can reformulate rigid O-unification as follows. Given a term rewriting system $\{t_i \to u_i \mid i \in [1..n]\}$, and a pair of terms v and w, decide if there exists a substitution θ such that $\theta(v) \to^* \theta(w)$ using the ground rewriting system $\{\theta(t_i) \to \theta(u_i) \mid i \in [1..n]\}$.

It is easy to prove decidability of rigid O-unification from decidability of rigid E-unification, and to prove undecidability of *simultaneous* rigid O-unification from undecidability of *simultaneous* rigid E-unification.

Proposition 2. *Rigid E-unification is reducible to rigid O-unification. Simultaneous rigid E-unification is reducible to simultaneous rigid E-unification.*

Proof: Replace every rigid equation $\bigwedge_{i\in[1..n]} t_i \cong u_i \vdash_\forall v \cong w$ by the rigid inclusion $\bigwedge_{i\in[1..n]}(t_i \subseteq u_i \wedge u_i \subseteq t_i) \vdash_\forall v \subseteq w$ ■

3 Reducing Simultaneous Rigid E-Unification to Second-Order Unification

In this section we reduce simultaneous rigid O-unification to second-order unification where second-order typed variables are restricted to occur at most twice in the unification problem. This reduction is based in the following main lemma.

[3] Requiring θ to be ground is not relevant since, if there exist a non-ground solution, then there exists also a ground solution.

Lemma 1 (Main Lemma). *The rigid equation over the signature* $\langle \Sigma, \mathcal{X} \rangle$

$$t_1 \subseteq u_1 \wedge \cdots \wedge t_m \subseteq u_m \vdash_\forall v \subseteq w \tag{1}$$

has a solution if, and only if, the following second-order equation

$$F(a(b, v), u_1, \ldots, u_m) \stackrel{?}{=} a(F(b, t_1, \ldots, t_m), w) \tag{2}$$

together with the following set of equations

$$\left.\begin{array}{l} X \stackrel{?}{=} G_x(f_1(\overrightarrow{Y_x}), \ldots, f_N(\overrightarrow{Y_x})) \\ b \stackrel{?}{=} G_x(b, \ldots, b) \end{array}\right\} \quad \forall X \in \mathcal{X} \,.\, \exists i \in [1..m] \,.\, u_i = X \tag{3}$$

have a solution.
Where we have assumed $\overrightarrow{Y_x} \notin \mathcal{X}$ *are lists of first-order variables of the appropriate length,* $a, b \notin \Sigma$ *and* $\Sigma = \{f_1, \ldots, f_N\}$ *is a finite signature.*[4]

Example 1. The following rigid inclusion $c \subseteq X \vdash_\forall f(c, c) \subseteq f(d, e)$, where $\Sigma = \{c, d, e, f\}$ and $\mathcal{X} = \{X\}$, is solvable if, and only if, the following set of second-order equations is solvable.

$$\begin{array}{l} F(a(b, f(c, c)), X) \stackrel{?}{=} a(F(b, c), f(d, e)) \\ X \stackrel{?}{=} G_x(c, d, e, f(Y^1_x, Y^2_x)) \\ b \stackrel{?}{=} G_x(b, b, b, b) \end{array}$$

In this case both systems are unsolvable. However, notice that the first equation $F(a(b, f(c, c)), X) \stackrel{?}{=} a(F(b, c), f(d, e))$ alone, has a solution

$$[F \mapsto \lambda x, y \,.\, y][X \mapsto a(c, f(d, e))]$$

To avoid this problem with variables occurring as the right-hand side of a premise, like in this case X, we require equations (3).

Proof of Main Lemma:
Implication $\Rightarrow$

Let θ be a solution of the rigid equation (1). Without lose of generality, we can assume that θ is ground and the signature Σ only contains constant symbols from equation (1). We can derive $\theta(v) \subseteq \theta(w)$ from $\bigwedge_{i \in [1..m]} \theta(t_i) \subseteq \theta(u_i)$ using only reflexivity, transitivity and monotonicity inference rules for the $\subseteq$ binary relation.

Using Birkhoff's theorem, we can prove that there exist a sequence of terms $s_1, \ldots, s_{k+1}$ such that $\theta(v) = s_1$ and $\theta(w) = s_{k+1}$ and for any $j \in [1..k]$ there exist an $i_j \in [1..m]$ and a position p_j of s_j such that

$$\begin{array}{l} s_{j+1} = s_j[\theta(u_{i_j})]_{p_j} \\ s_j|_{p_j} = \theta(t_{i_j}) \end{array}$$

[4] We can consider Σ as the set of constants $f_1, \ldots, f_N$ occurring in the original rigid E-unification problem.

i.e. we can rewrite s_j into s_{j+1} using $t_{i_j} \to u_{i_j}$ as a rewriting rule at position p_j.

Define the second-order substitution σ as follows:

$$\sigma(X) = \theta(X) \quad \text{for } X \in \mathcal{X}$$
$$\sigma(F) = \lambda x_0, x_1, \ldots, x_m \,.\, a(\ldots a(a(x_0, s_1[x_{i_1}]_{p_1}), s_2[x_{i_2}]_{p_2}) \ldots, s_k[x_{i_k}]_{p_k})$$

It is a straightforward exercise to prove that this substitution σ is a solution of the second-order equation (2).

For any variable X satisfying $\exists j \in [1..m] \,.\, u_i = X$ define σ_x as follows. Let $f_k \in \Sigma$ be the constant such that $\theta(X) = f_k(s_1, \ldots, s_p)$, for some terms $s_1, \ldots, s_p$, where $p = \text{arity}(f_k)$, (regard that θ is ground and the signature Σ only contains constant symbols from equation (1)), then

$$\sigma_x(\overrightarrow{Y_x}) = \overrightarrow{s}$$
$$\sigma_x(G_x) = \lambda x_0, x_1, \ldots, x_N \,.\, x_k$$

It is also straightforward to prove that $\sigma_x \circ \sigma$ satisfies equation (3) for X.

As far as these substitutions σ and σ_x, for any X, have disjoint domains, the composition of all them $\sigma \circ \circ_{x \in \mathcal{X}} \sigma_x$ solves all the second-order equations (2) and (3).

Implication $\Leftarrow$

Suppose that equation (2) is solvable. We can only apply the imitation or projection rules, and by completeness of the pre-unification procedure, one of the two problems that we obtain has to be solvable.

If we apply the projection rule, it has to be necessarily $[F \mapsto \lambda x_0, \ldots, x_m \,.\, x_0]$. Any other projection function would lead to $u_i \stackrel{?}{=} a(t_i, w)$, and this equation has no solution unless u_i contains a (which is not the case because $a \notin \Sigma$) or u_i is a variable. If $u_i = X$, we have equations (3) written for X. The only possible solutions for the second one of these equations $b \stackrel{?}{=} G_x(b, \ldots, b)$ are $[G_x \mapsto \lambda x_1, \ldots, x_n \,.\, x_k]$, for some $k \in [1..N]$, and $[G_x \mapsto \lambda x_1, \ldots, x_n \,.\, b]$. Applying any of this substitutions to the rest of equations $X \stackrel{?}{=} a(t_i, w)$ and $X \stackrel{?}{=} G_x(f_1(\overrightarrow{Y_x}), \ldots, f_N(\overrightarrow{Y_x}))$ results in the following unsolvable problems $\{X \stackrel{?}{=} a(t_i, w), X \stackrel{?}{=} f_k(\overrightarrow{Y_x})\}$, for some $k \in [1..N]$, or $\{X \stackrel{?}{=} a(t_i, w), X \stackrel{?}{=} b\}$. Therefore, we can conclude that, after instantiating F by the only possible projection function, we obtain $a(b, v) \stackrel{?}{=} a(b, w)$. Now, by simplification we obtain $v \stackrel{?}{=} w$.

If we apply the imitation rule, we obtain:

$$F_1(a(b, v), u_1, \ldots, u_m) \stackrel{?}{=} a(F_1(b, t_1, \ldots, t_m), F_2(b, t_1, \ldots, t_m))$$
$$F_2(a(b, v), u_1, \ldots, u_m) \stackrel{?}{=} w$$

where F_1 and F_2 are both fresh variables.

If this system is solvable, then the first equation –which is quite similar to the original one– has to be also solvable. We can repeat the same argument for this equation. Iterating this argument, we can conclude that: there exist a $k \geq 0$ such that, after applying k times the imitation rule to the first equation, and

later the projection and the simplification rules, the system we get is solvable. The system will be:

$$\begin{aligned}
&v \stackrel{?}{=} F_1(b, t_1, \ldots, t_m)\\
&F_1(a(b,v), u_1, \ldots, u_m) \stackrel{?}{=} F_2(b, t_1, \ldots, t_m)\\
&F_2(a(b,v), u_1, \ldots, u_m) \stackrel{?}{=} F_3(b, t_1, \ldots, t_m)\\
&\cdots\\
&F_k(a(b,v), u_1, \ldots, u_m) \stackrel{?}{=} w
\end{aligned}$$

for $k > 0$, and $u \stackrel{?}{=} w$ for $k = 0$.

Solvability of one of these system ensures that there exist a ground substitution σ such that $\sigma(v)$ can be rewritten into $\sigma(F_1(a(b,v), u_1, \ldots, u_m))$ in one parallel rewriting step, and $\sigma(F_1(a(b,v), u_1, \ldots, u_m))$ can be rewritten into $\sigma(F_2(a(b,v), u_1, \ldots, u_m))$, etc. using ground rewriting rules: $\sigma(b) \to \sigma(a(b,v))$, $\sigma(t_1) \to \sigma(u_1)$, ... , $\sigma(t_m) \to \sigma(u_m)$. Again, Birkhoff's theorem proves that we can deduce:

$$\sigma(b) \subseteq \sigma(a(b,v)) \wedge \sigma(t_1) \subseteq \sigma(u_1) \wedge \cdots \wedge \sigma(t_m) \subseteq \sigma(u_m) \;\vdash\; \sigma(v) \subseteq \sigma(w)$$

in first-order logic with a monotonic pre-order relation $\subseteq$. Therefore, the rigid equation:

$$b \subseteq a(b,v) \wedge t_1 \subseteq u_1 \wedge \cdots \wedge t_m \subseteq u_m \;\vdash_\forall\; v \subseteq w$$

has a solution. As far as a and b do not occur in t_i, u_i, v and w, we do not need to assume $b \subseteq a(b,v)$ in that derivation. Therefore, we can also prove that the rigid equation (1) has a solution. ■

Theorem 1. *There is an effective method that reduces simultaneous rigid E-unification to second-order unification, where second-order variables are restricted to occur at most twice in a unification problem and no equation contains more than one second-order variable.*

Proof: We have already seen that simultaneous rigid E-unification is reducible to simultaneous rigid O-unification. Now, lemma 1 can be easily extended to reduce simultaneous rigid O-unification to second-order unification.

Suppose we have a system of n rigid equations, over a first-order signature $\langle \Sigma, \mathcal{X} \rangle$,

$$t_1^i \subseteq u_1^i \wedge \cdots \wedge t_{m_i}^i \subseteq u_{m_i}^i \vdash_\forall v^i \subseteq w^i$$

for $i \in [1..n]$.

We define a new second-order signature

$$\langle \Sigma \cup \{a, b\}, \mathcal{X} \cup \bigcup_{\substack{x \in \mathcal{X} \\ j \in [1..N]}} \{Y_x^j\} \cup \bigcup_{i \in [1..n]} \{F^i\} \cup \bigcup_{x \in \mathcal{X}} \{G_x\} \rangle$$

where, apart from the constants and variables we already had, we have introduced a new constant symbol b, a new binary function symbol a, N first-order

variables for each variable $X \in \mathcal{X}$, where $N = \max\{arity(f) \mid f \in \Sigma\}$, a second-order variable F^i, with arity $m_i + 1$, for each rigid equation of the system, and a second-order variable G_x with arity cardinality of Σ, for each variable $X \in \mathcal{X}$.

We can effectively construct a second-order unification problem, containing the following second-order equation for each rigid equation:

$$F^i(a(b, v^i), u^i_1, \ldots, u^i_{m_i}) \stackrel{?}{=} a(F^i(b, t^i_1, \ldots, t^i_{m_i}), w^i)$$

and the following equations

$$X \stackrel{?}{=} G_x(f_1(\overrightarrow{Y_x}), \ldots, f_N(\overrightarrow{Y_x}))$$
$$b \stackrel{?}{=} G_x(b, \ldots, b)$$

for any variable $X \in \mathcal{X}$ satisfying $\exists i \in [1..n] \,.\, \exists j \in [1..m_i] \,.\, u^i_j = X$.

An extension of main lemma can be used to prove the equivalence between this system and the original simultaneous rigid equations. Notice that any second-order variable F^i or G_x in the equations occurs only twice and there are not equations containing more than one second-variable. ■

Since there exist a reduction [DV95] from second-order unification to simultaneous rigid E-unification we have the following corollary.

Corollary 1. *Second-order unification, simultaneous rigid E-unification, and second-order unification where second-order variables are restricted to occur at most twice and equations to do not contain more than one second-order variable, are all three equivalent.*
The second-order unification problem is undecidable, even if we restrict second-order variables to occur at most twice, and equations to do not contain more than one second-order variable.

4 Decidability Results

In this section we prove decidability of second-order unification problems only containing one second-order variable, which occurs at most twice. The impossibility to prove this result for more than one variable suggested us how to prove undecidability (in the previous section). It also helped us establish a relationship between second-order unification and simultaneous rigid E-unification. Additionally, we also prove decidability for problems not containing any repeated variable.

If we compare second-order unification rules with word unification rules, in section 2, at first sight it seems that the two-occurrences restriction is going to carry over, like in the word unification case. The simplification and projection rules always decrease the size of a problem. However, in this case, application of imitation rule can increase the size of a problem, even if we restrict variables to occur at most twice.

For instance, if we apply imitation rule to

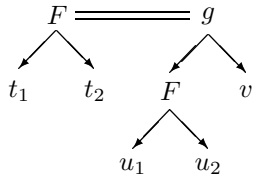
$$F(t_1, t_2) \stackrel{?}{=} g(F(u_1, u_2), v)$$

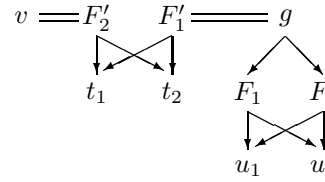
we obtain a bigger problem (in term of number of symbols)

$$F'_1(t_1, t_2) \stackrel{?}{=} g(F'_1(u_1, u_2), F'_2(u_1, u_2))$$
$$F'_2(t_1, t_2) \stackrel{?}{=} v$$

Moreover, since some terms are duplicated (like t_1, t_2, u_1, u_2), second-order variables of these terms may occur now more than twice!

We can overcome the second problem by assigning a directed acyclic graph (DAG) to each problem to avoid duplication of terms. In our example we would have:





$F(t_1, t_2) \stackrel{?}{=} g(F(u_1, u_2), v)$

$F'_1(t_1, t_2) \stackrel{?}{=} g(F'_1(u_1, u_2), F'_2(u_1, u_2))$
$F'_2(t_1, t_2) \stackrel{?}{=} v$

Then, we can define the *size of a problem* as a pair *(number of constant occurrences, number of variable occurrences)* of its assigned DAG. We compare these pairs using a lexicographic order. Since the simplification rule always removes two constant occurrences, it always decreases the size of the problem. The projection rule removes a variable occurrence and does not increase the number of constant occurrences. However, the imitation rule may increase the number of variable occurrences (although, if no variable occurs more than twice, it never increases the number of constant occurrences). This proves the following lemma.

Lemma 2. *Any infinite transformation sequence contains infinitely many imitation steps.*

To characterise non-terminating transformation sequences we have to study the imitation rule in detail. When we apply the imitation rule to a rigid-flexible equation $F(t_1, \ldots, t_n) \stackrel{?}{=} g(u_1, \ldots, u_m)$, the occurrence of g on the right hand side of the equation is removed, i.e. the equation is replaced by new equations $F'_i(t_1, \ldots, t_n) \stackrel{?}{=} u_i$ for $i \in [1..m]$. And, if there exists another occurrence of F in another term, an occurrence of g is added to this term. Thus, we can see the imitation rule as *moving* constant occurrences from one place to another. This image can help to characterise non-terminating sequences as follows.

Definition 1. *We say that a unification problem is in* **normal form** *if it does not contain any rigid-rigid equation. Notice that, given a solvable unification problem, we can always find an equivalent problem in normal form by repeatedly applying simplification rule.*

We say that two variables F and G are **equivalent** *in an unification problem S, noted $F \cong G$, if for some substitution θ, the normal form of $\theta(S)$ contains a flexible-flexible equation of the form $F(t_1, \ldots, t_n) \stackrel{?}{=} G(u_1, \ldots, u_m)$.*
We say that a variable F is **connected with** *another variable G in a unification problem S, noted $F \rightsquigarrow G$, if for some substitution θ, the normal form of $\theta(S)$ contains an equation of the form $F(t_1, \ldots, t_n) \stackrel{?}{=} v$, where v contains the variable G, and it is not in the head of v.*
Let $\stackrel{\cong}{\rightsquigarrow}$ denote the relation $\cong^* \circ \rightsquigarrow$. We say that a unification problem contains a **variable cycle** *if there is a non-empty sequence of variables such that $F_1 \stackrel{\cong}{\rightsquigarrow} F_2 \stackrel{\cong}{\rightsquigarrow} \cdots \stackrel{\cong}{\rightsquigarrow} F_n$.*

In our example $F(t_1, t_2) \stackrel{?}{=} g(F(u_1, u_2), v)$, we have $F \rightsquigarrow F$, therefore, it contains a variable cycle.

Theorem 2. *Any infinite transformation sequence is generated by a problem containing a variable cycle.*
Therefore, it is decidable whether a second-order unification problem not containing variable cycles has a unifier.

Proof: By lemma 2 we know that any infinite sequence contains infinitely many imitation steps. Since initially there are finitely many variables, and when we instantiate one variable we only introduce finitely many new fresh variables, we can conclude that, some variable F of the original problem is involved in an *infinite sequence of chained imitation steps*:

$$\cdots [F \mapsto \lambda \overline{x} \,.\, g_1(\ldots, F_1(\overline{x}), \ldots)] \cdots [F_1 \mapsto \lambda \overline{x} \,.\, g_2(\ldots, F_2(\overline{x}), \ldots)] \cdots$$

Assume F is one of the maximal (w.r.t. the relation $\stackrel{\cong}{\rightsquigarrow}$) variables involved in one of such chained sequences. This is always possible unless the relation is cycling.

Firstly, we will prove that some variable G satisfying $F \stackrel{\cong}{\rightsquigarrow} G$ is also involved in one of such infinite sequences of chained imitation steps.

At some point of the transformation sequence, the problem contains, at least, one rigid-flexible pair $F(\ldots) \stackrel{?}{=} g_1(t_1, \ldots, t_m)$. Otherwise, the imitation step $[F \mapsto \lambda \overline{x} \,.\, g_1(\ldots F_1(\overline{x}) \ldots)]$ would never been applied. By applying this imitation step to this problem, we replace this equation by a finite set of equations containing $F_1(\ldots) \stackrel{?}{=} \rho(t_i)$ for some $i \in [1..m]$. Notice that an occurrence of the function symbol g_1 is removed when we replace $g_1(t_1, \ldots, t_m)$ by $\rho(t_i)$ on the right-hand side of equations. Therefore, since there are finitely many occurrences of function symbols in $g_1(t_1, \ldots, t_m)$, we can not repeat this process infinitely many times, unless some variable G occurring in t_i is also involved in an infinite sequence of chained imitation steps. If G is a variable of the original problem, we have $F \rightsquigarrow G$, and the work is done. Otherwise, let G' be the variable of the original problem that originates the chained sequence where G is involved. We can prove that initially there is an equation containing both F and G', and either $F \rightsquigarrow G'$, $F \cong G'$ or $G' \rightsquigarrow F$. In the first case the work is done. The last case is not possible because we have assumed that F is maximal. In the second case we can repeat the same reasoning for G'. At some point we have to find a

variable G'' such that $F \cong G' \overset{\cong}{\rightsquigarrow} G''$. Otherwise it is not possible to have $F \rightsquigarrow G$ at some point of the transformation sequence.

Now we can repeat the same argument for G, or G' or G''. Since originally there are finitely many variables, this process allows us to construct a cycle $F \overset{\cong}{\rightsquigarrow}^* H \overset{\cong}{\rightsquigarrow}^* \circ \overset{\cong}{\rightsquigarrow} H$ for some variable H of the original problem. Moreover H is involved in an infinite sequence of chained imitation steps. ■

A direct consequence of this theorem is the following decision result.

Corollary 2. *It is decidable whether a second-order unification problem, where no second-order variable occurs more than once, has a unifier.*

Proof: Ii is not difficult to prove that, if no variable is repeated, we can not have any cycle.

We can also prove this result directly. If no variable is repeated, no transformation rule can increase the size of the problem. However, we still have to prove that no transformation rule can duplicate a variable occurrence. This is true if we represent the unification problem as a DAG. ■

If we have multiple occurrences of a variable, we have to deal with infinite transformation sequences. This does not seems easy. We only have been able to do that when the problem only contains a second-order variable and this variable only occurs twice.

Theorem 3. *It is decidable whether a second-order unification problem, containing a single second-order variable and where this variable only occurs twice, has a unifier.*

Proof: The only possible cycle $F \rightsquigarrow F$ is generated if we have an equation $\lambda\overline{x}\,.\,F(t_1,\ldots,t_n) \stackrel{?}{=} v$ where F occurs in v.

For simplicity, we will assume that only one function symbol g occurs between the root of v and F. Without loss of generality, we will assume that F occurs in the first argument of v. We can have other more complex situations, but they can also be proved to be decidable using the same main ideas. Under these simplifications, we only need to consider the following equation.

$$F(t_1,\ldots,t_n) \stackrel{?}{=} g(F(u_1,\ldots,u_n),v_1,\ldots,v_m)$$

We can repeat the same argument as in the proof of lemma 1. If this equation has a solution, since it is a rigid-flexible equation, we can obtain another solvable system after applying the imitation rule k many times to the first equation and later the projection rule, for some $k \geq 0$. If we only apply imitation rule, we can generate an infinite transformation sequence. However, if the problem has a solution, there is a *finite* sequence of transformations leading to a set with only flexible-flexible equations. Therefore, at some point we have to apply projection rule to the first equation. The problem is that we can not conjecture when!

After these k many imitation steps, and a projection step, we get a system which is equivalent to the following one.

$$
\begin{array}{l}
t_i \stackrel{?}{=} g(u_i, X_1, \ldots, X_m) \\
X_1 \stackrel{?}{=} F_0^{(p)}(u_1, \ldots, u_n) \\
\ldots \\
X_m \stackrel{?}{=} F_m^{(p)}(u_1, \ldots, u_n) \\
F_1^{(p)}(t_1, \ldots, t_n) \stackrel{?}{=} F_1^{(p-1)}(u_1, \ldots, u_n) \\
\ldots \\
F_m^{(p)}(t_1, \ldots, t_n) \stackrel{?}{=} F_m^{(p-1)}(u_1, \ldots, u_n) \\
F_1^{(p-1)}(t_1, \ldots, t_n) \stackrel{?}{=} F_1^{(p-2)}(u_1, \ldots, u_n) \\
\ldots \\
F_m^{(p-1)}(t_1, \ldots, t_n) \stackrel{?}{=} F_m^{(p-2)}(u_1, \ldots, u_n) \\
\ldots \\
F_1^{(1)}(t_1, \ldots, t_n) \stackrel{?}{=} v_1 \\
\ldots \\
F_m^{(1)}(t_1, \ldots, t_n) \stackrel{?}{=} v_m
\end{array}
$$

for some $i \in [1..n]$.

Applying the same ideas as in section 3, we can prove that solvability of this system is equivalent to solvability of the following rigid equation:

$$\sigma(t_1) \subseteq \sigma(u_1) \wedge \cdots \wedge \sigma(t_n) \subseteq \sigma(u_n) \;\vdash_\forall\; \sigma(v_1) \subseteq \sigma(X_1) \wedge \cdots \wedge \sigma(v_m) \subseteq \sigma(X_m)$$

for some σ being unifier of $t_i \stackrel{?}{=} g(u_i, X_1, \ldots, X_m)$ for some $i \in [1..n]$. There are finitely many of such unifiers. The problem has been reduced to solvability of finitely many instances of a rigid equation, which is decidable.

For more complex cycle situations (not considered in this proof) we get a similar rigid equation. ■

5 Conclusions and Further Work

Since Goldfarb proved the undecidability of the second-order unification problem [Gol81], very few decidable and undecidable subclasses of second-order unification problems have been found. Here we have characterised decidability for classes defined in terms of number of occurrences per variable.

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

In [Lev96] we proved that *linear* second-order unification is decidable when no second-order variable occurs more than twice. Here, we have proved that, under this same restriction, second-order unification is undecidable. This establishes a clear difference between this two apparently similar problems. Notice that second-order unification is undecidable, whereas linear second-order unification has been conjectured to be decidable.

Acknowledgements

I would like to acknowledge M. Bonet, A. Rubio, M. Villaret and all the anonymous referees of this paper for their comments and support, and specially to R. Nieuwenhuis for suggesting me the possible relationship between the problem I *tried* to prove decidable and the simultaneous rigid E-unification problem.

References

[Com93] H. Comon. Completion of rewrite systems with membership constraints. Technical report, CNRS and LRI, Université de Paris Sud, 1993.

[DV95] A. Degtyarev and A. Voronkov. Reduction of second-order unification to simultaneous rigid E-unification. Technical Report 109, Computer Science Department, Uppsala University, 1995.

[DV96] A. Degtyarev and A. Voronkov. The undecidability of simultaneous rigid E-unification. *Theoretical Computer Science*, 166(1-2):291–300, 1996.

[Far88] W. M. Farmer. A unification algorithm for second-order monadic terms. *Annals of Pure and Applied Logic*, 39:131–174, 1988.

[GNPS88] J. H. Gallier, P. Narendran, D. Plaisted, and W. Snyder. Rigid E-unification is NP-complete. In *Proc. IEEE Conf. on Logic in Computer Science, LICS'88*, pages 338–346, 1988.

[Gol81] W. D. Goldfarb. The undecidability of the second-order unification problem. *Theoretical Computer Science*, 13:225–230, 1981.

[GRS87] J. H. Gallier, S. Raatz, and W. Snyder. Theorem proving using rigid E-unification: Equational matings. In *Proc. IEEE Conf. on Logic in Computer Science, LICS'87*, pages 338–346, 1987.

[GS90] J. H. Gallier and W. Snyder. Designing unification procedures using transformations: A survey. *Bulletin of the EATCS*, 40:273–326, 1990.

[Hue75] G. Huet. A unification algorithm for typed λ-calculus. *Theoretical Computer Science*, 1:27–57, 1975.

[Lev96] J. Levy. Linear second-order unification. In *7th Int. Conf. on Rewriting Techniques and Applications, RTA'96*, volume 1103 of *LNCS*, pages 332–346, New Jersey, USA, 1996.

[Mak77] G. S. Makanin. The problem of solvability of equations in a free semigroup. *Math. USSR Sbornik*, 32(2):129–198, 1977.

[Pie73] T. Pietrzykowski. A complete mechanization of second-order logic. *J. of the ACM*, 20(2):333–364, 1973.

[Sch91] K. U. Schulz. Makanin's algorithm, two improvements and a generalization. Technical Report CIS-Bericht-91-39, Centrum für Informations und Sprachverarbeitung, Universität München, 1991.

[Sch97] A. Schubert. Second-order unification and type inference for church-style polymorphism. Technical Report TR 97-02(239), Institute of Informatics, Warsaw University, 1997.

[SS95] M. Schmidt-Schauß. Unification of stratified second-order terms. Technical Report 12/94, Johan Wolfgang-Goethe-Universität, Frankfurt, Germany, 1995.

[Vea98] M. Veanes. The relation between second-order unification and simultaneous rigid E-unification. Technical Report MPI-I-98-2-005, Max-Planck Institut für Informatik, 1998.