Sequence Unification Through Currying*

Temur Kutsia[1], Jordi Levy[2], and Mateu Villaret[3]

[1] Research Institute for Symbolic Computation (RISC),
Johannes Kepler University of Linz, Linz, Austria.
http://www.risc.uni-linz.ac.at/people/tkutsia/
[2] Artificial Intelligence Research Institute (IIIA),
Spanish Council for Scientific Research (CSIC), Barcelona, Spain.
http://www.iiia.csic.es/~levy
[3] Departament d'Informàtica i Matemàtica Aplicada (IMA),
Universitat de Girona (UdG), Girona, Spain.
http://ima.udg.es/~villaret

Abstract. Sequence variables play an interesting role in unification and matching when dealing with terms in an unranked signature. Sequence Unification generalizes Word Unification and seems to be appealing for information extraction in XML documents, program transformation, and rule-based programming.
In this work we study a relation between Sequence Unification and another generalization of Word Unification: Context Unification. We introduce a variant of Context Unification, called Left-Hole Context Unification that serves us to reduce Sequence Unification to it: We define a partial currying procedure to translate sequence unification problems into left-hole context unification problems, and prove soundness of the translation. Furthermore, a precise characterization of the shape of the unifiers allows us to easily reduce Left-Hole Context Unification to (the decidable problem of) Word Unification with Regular Constraints, obtaining then a decidability proof for an extension of Sequence Unification.

1 Introduction

In this work we study a relation between Sequence and Context Unification. Both problems are generalizations of Word Unification [7, 13, 22, 26, 30]. Word Unification is the problem of solving equations between terms build up from letters and word variables. A solution of a word equation is a mapping from variables to words that when applied to both sides of the equation the result is the same word.

Sequence Unification is the problem of solving equations between terms built up using an unranked signature (aka flexible arity, or variadic function symbols)

* This research has been partially funded by the CICYT research projects iDEAS (TIN2004-04343) and Mulog (TIN2004-07933-C03-01), by the Austrian Science Foundation (FWF) under Project SFB F1302, and by the EC Framework 6 Programme for Integrated Infrastructures Initiatives under the project SCIEnce (026133).

and sequence and individual variables. Sequence variables are instantiated with finite sequences of terms, while individual variables instantiate to a single term. Sequence Unification is decidable and infinitary [15, 16].

Solving equations with sequence variables has quite a broad range of applications. The rule-based programming language of Mathematica [31] relies on a pattern matching mechanism, which supports sequence variables and flexible arity function symbols. It can do matching modulo certain equational theories as well. Solving equations with sequence variables form a basis for schema transformation operations [3, 27] used in synthesis and transformation of logic programs. Other applications include knowledge engineering and artificial intelligence [8, 11, 12], automated reasoning [2, 9, 25], rewriting [10], functional logic programming [1]. The ISO standard proposal for Common Logic [6] has notation for sequence variables (called there sequence markers). Recently there have been developments in XML querying and transformation that model XML documents with terms over an unranked signature and use sequence matching and unification techniques [15] for querying, transforming, and verifying them [4, 5]. Obviously, we can not give an exhaustive overview of all the applications here.

Context Unification is the problem of solving equations between terms built up using ranked signatures and with first-order and context variables. The latter occur as monadic function symbols and denote contexts, i.e. terms with exactly one hole. When the ranked signature considered is restricted to not contain symbols of arity greater than one, the problem is equivalent to Word Unification. When allowing one single binary symbol, its decidability is still unknown [21]. Nevertheless several fragments and variants are known to be decidable [18, 19, 28]. The main application field of Context Unification is computational linguistics, mainly in compositional semantics of natural language [14, 19, 24].

Combining sequence and context variables in a single framework and equipping it with regular constraint solving methods makes the framework more flexible, with many potential applications [17, 23].

The goal of this paper is to look in depth into relations between Sequence and Context Unification. Throughout curryfication we define a translation from sequence unification problems into context unification problems over a signature consisting of constants and a single binary function symbol @ (curried context unification problems), in addition, sequence variables are "encoded" into context variables while individual variables become first-order variables. The translation preserves solvability in one direction: If the sequence unification problem is solvable, then the corresponding context unification problem is solvable. To preserve solvability in the other direction, we have to restrict possible solutions of the curried context unification problems, which leads to a new variant of Context Unification that we call Left-Hole Context Unification. We prove that Left-Hole Context Unification is a decidable variant of Context Unification. We do it by reducing Left-Hole Context Unification to Word Unification with Regular Constraints that is known to be decidable [30]. The reduction transforms context equations into word equations on the postorder traversal of the terms. Regu-

lar constraints are required to filter the solutions of the word equations that correspond to traversals of terms. This reduction is based in some ideas of [20].

With these reductions we get a new decidability proof for Sequence Unification, easier than the one in [16]. In addition we also get decidability for an extension of Sequence Unification. Moreover, this translation also allows us to use the complexity results for context matching of [29] to characterize complexity of sequence matching.

The paper proceeds as follows: Section 2 defines the two main problems: Sequence and Context Unification, Section 3 introduces the currying encoding and shows its soundness, in Section 4 decidability of Left-Hole Context Unification is shown, Section 5 discusses some extensions of Sequence Unification thanks to the currying process, Section 6 is the conclusion.

2 Preliminary Definitions

2.1 Unranked Signatures

Given an unranked signature Σ_{U} (i.e., a finite set of function symbols that have flexible arity), a countable set of individual variables $\mathcal{V}_{\mathsf{I}}$, and a countable set of sequence variables $\mathcal{V}_{\mathsf{S}}$, we define *unranked terms over* Σ_{U} *and* $\mathcal{V} = \mathcal{V}_{\mathsf{I}} \cup \mathcal{V}_{\mathsf{S}}$ by the following grammar:

$$r ::= v \mid V \mid f(r_1, \ldots, r_n)$$

where $v \in \mathcal{V}_{\mathsf{I}}$, $V \in \mathcal{V}_{\mathsf{S}}$, $f \in \Sigma_{\mathsf{U}}$, and $n \geq 0$. The sets Σ_{U}, $\mathcal{V}_{\mathsf{I}}$ and $\mathcal{V}_{\mathsf{S}}$ are mutually disjoint. We will abbreviate terms of the form $f()$ by f. The set of unranked terms over Σ_{U} and $\mathcal{V}$ is denoted by $\mathcal{T}(\Sigma_{\mathsf{U}}, \mathcal{V})$, or simply by $\mathcal{T}_{\mathsf{U}}$ when the signature and the set of variables are unimportant. The letters f, g, a and b will be used for function symbols, v and u for individual variables, V and U for sequence variables, w for individual or sequence variables, and r and l for unranked terms. We call unranked terms from $\mathcal{T}(\Sigma_{\mathsf{U}}, \mathcal{V}) \setminus \mathcal{V}_{\mathsf{S}}$ the *individual terms*.

A *substitution for individual and sequence variables* (*IS-substitution* for short), is a mapping from individual variables to individual terms, and from sequence variables to finite sequences of unranked terms such that all but finitely many individual variables are mapped to themselves, and all but finitely many sequence variables are mapped to themselves considered as singleton sequences.[1] We use the Greek letters σ and ϑ to denote IS-substitutions.

The *composition* of two IS-substitutions σ and ϑ, written as $\sigma \circ \vartheta$, is defined by $(\sigma \circ \vartheta)(r) = \sigma(\vartheta(r))$. Given an IS-substitution σ, we represent it as $[v_1 \mapsto \sigma(v_1), \ldots, v_n \mapsto \sigma(v_n), V_1 \mapsto \sigma(V_1), \ldots, V_m \mapsto \sigma(V_m)]$ where v's and V's are all those variables for which $\sigma(v) \neq v$ and $\sigma(V) \neq V$.

The application of an IS-substitution σ to an unranked term r, denoted $\sigma(r)$, is defined by

$$\sigma(r) := \begin{cases} \sigma(w) & \text{if } r = w \\ f(\sigma(r_1), \ldots, \sigma(r_n)) & \text{if } r = f(r_1, \ldots, r_n) \end{cases}$$

[1] We do not distinguish between a singleton sequence and its sole member.

Similarly, σ can be applied to a sequence of unranked terms $\langle r_1, \ldots, r_n \rangle$: $\sigma(\langle r_1, \ldots, r_n \rangle) = \langle \sigma(r_1), \ldots, \sigma(r_n) \rangle$. For the sake of readability, we put sequences between angular brackets, that are later flattened: if $\sigma = [u \mapsto a, V \mapsto \langle f(b), c \rangle]$ then $\sigma(f(u, V)) = f(a, \langle f(b), c \rangle) = f(a, f(b), c)$. We call $\sigma(r)$ (resp. $\sigma(\langle r_1, \ldots, r_n \rangle)$) an *instance* of r (resp. of $\langle r_1, \ldots, r_n \rangle$) under σ. Note that the set of unranked terms is not closed under IS-substitution application: An instance of a sequence variable is an unranked term sequence that in general is not an unranked term. However, an instance of an individual term is always an individual term. An IS-substitution σ_1 is *more general* than σ_2 with respect to a set of variables W, written $\sigma_1 \preceq^W \sigma_2$, if there exists ϑ such that $(\vartheta \circ \sigma_1)(w) = \sigma_2(w)$, for each $w \in W$.

A *sequence unification problem* (SU problem) is a set of *equations* (unoriented pairs) of individual terms, denoted $\{l_1 \stackrel{?}{=} r_1, \ldots, l_n \stackrel{?}{=} r_n\}$. IS-Substitutions extend to equations and unification problems: $\sigma(l_1 \stackrel{?}{=} r_1) = \sigma(l_1) \stackrel{?}{=} \sigma(r_1)$ and $\sigma(\{e_1, \ldots, e_n\}) = \{\sigma(e_1), \ldots, \sigma(e_n)\}$. A *unifier* of a sequence unification problem Γ is an IS-substitution σ such that $\sigma(l) = \sigma(r)$ for each $l \stackrel{?}{=} r \in \Gamma$, and Γ is *solvable* if it has a unifier. A unifier σ of Γ is called *ground*, if $\sigma(\Gamma)$ contains no variables. A unifier σ_1 of Γ is *more general* than another σ_2, if $\sigma_1 \preceq^{vars(\Gamma)} \sigma_2$. A unifier σ is *most general*, if any other unifier σ' satisfying $\sigma' \preceq^{vars(\Gamma)} \sigma$ also satisfies $\sigma \preceq^{vars(\Gamma)} \sigma'$.

2.2 Ranked Signatures

A ranked signature Σ_{R} is a finite set of fixed arity function symbols. We assume that Σ_{R} contains the 0-ary symbol $\bullet$, called the *hole*. Given Σ_{R}, a countable set of first-order variables $\mathcal{X}_{\mathsf{F}}$, and a countable set of context variables $\mathcal{X}_{\mathsf{C}}$, we define *ranked terms over* Σ_{R} *and* $\mathcal{X} = \mathcal{X}_{\mathsf{F}} \cup \mathcal{X}_{\mathsf{C}}$ by the following grammar:

$$t ::= x \mid X(t) \mid f(t_1, \ldots, t_n)$$

where $x \in \mathcal{X}_{\mathsf{F}}$, $X \in \mathcal{X}_{\mathsf{C}}$, $f \in \Sigma_{\mathsf{R}}$ such that f is n-ary and with $n \geq 0$. When $n = 0$, we omit the parentheses and write just f. The sets Σ_{R}, $\mathcal{X}_{\mathsf{F}}$ and $\mathcal{X}_{\mathsf{C}}$ are mutually disjoint. *Constants* are 0-ary function symbols. The set of ranked terms over Σ_{R} and $\mathcal{X}$ is denoted by $\mathcal{T}(\Sigma_{\mathsf{R}}, \mathcal{X})$ or simply by $\mathcal{T}_{\mathsf{R}}$ when the signature and the set of variables are unimportant. A *context* is a ranked term with exactly one occurrence of the hole. A context C may be *applied* to a ranked term t, written $C[t]$, and the result is a ranked term over Σ_{R} and $\mathcal{X}$ obtained from C by replacing the hole with t. The letters x and y will be used for first-order variables, X and Y for context variables, z for first-order or context variables, a and b for constants, f for function symbols and s and t for ranked terms. The size of a term t, noted by $|t|$, is defined as its number of symbols.

A *substitution for first-order and context variables,* or an *FC-substitution* in short, is a mapping from first-order variables to hole-free ranked terms, and from context variables to contexts such that all but finitely many first-order variables are mapped to themselves, and all but finitely many context variables are mapped to themselves applied to the hole. We use the Greek letters φ and ρ to denote them. Composition is defined as above.

Given an FC-substitution φ, we represent it as $[x_1 \mapsto \varphi(x_1), \ldots, x_n \mapsto \varphi(x_n), X_1 \mapsto \varphi(X_1), \ldots, X_m \mapsto \varphi(X_m)]$, where x's are all first-order variables such that $\varphi(x) \neq x$, and X's are all context variables such that $\varphi(X) \neq X(\bullet)$.

The application of an FC-substitution φ to a ranked term t, denoted $\varphi(t)$, is defined by

$$\varphi(t) := \begin{cases} \varphi(x) & \text{if } t = x \\ \varphi(X)[\varphi(s)] & \text{if } t = X(s) \\ f(\varphi(s_1), \ldots, \varphi(s_n)) & \text{if } t = f(s_1, \ldots, s_n) \end{cases}$$

A *context unification problem* (CU problem) is a set of *equations* (unoriented pairs) of ranked *hole-free* terms, denoted $\{s_1 \stackrel{?}{\approx} t_1, \ldots, s_n \stackrel{?}{\approx} t_n\}$. A *unifier* of a context unification problem Δ is an FC-substitution φ such that $\varphi(s) = \varphi(t)$, for each $s \stackrel{?}{\approx} t \in \Delta$, and Δ is solvable, if it has a unifier.

The notions of a more general and most general FC-unifier are defined in the same way as for IS-unifiers.

3 Currying Terms

In this section we define the *currying* transformation that will serve us to transform sequence unification problems into (a variant of) context unification problems.

We firstly thought that this reduction was solvability preserving: a sequence unification problem is solvable, if, and only if, its transformation into a context unification problem is solvable. However, although the implication to the right is almost trivial (Lemma 2), while trying to prove the other direction shows us that there are solutions of the context unification problems that, when interpreted as solutions for sequence unification problems, are not valid. We find out what is the kind of solutions that we need to consider in order to get the left implication. This characterization leads us to make the reduction, not directly to Context Unification but to a variant of it, called *Left-Hole Context Unification*.

We assume that for each $f \in \Sigma_{\mathsf{U}}$ there exists a unique and distinct constant $a_f \in \Sigma_{\mathsf{R}} \setminus \{\bullet\}$. The set of these constants is denoted by $\Sigma_0^{\mathcal{C}}$. We also assume that Σ_{R} contains a binary function symbol @ and define $\Sigma_2^{\mathcal{C}} = \{@\}$. Then, the set $\Sigma_{\mathsf{U}}^{\mathcal{C}} = \Sigma_0^{\mathcal{C}} \cup \Sigma_2^{\mathcal{C}}$ is called the *curried signature* corresponding to Σ_{U}.

Similarly, we associate to each $v \in \mathcal{V}_{\mathsf{I}}$ a unique and distinct first-order variable $x_v \in \mathcal{X}_{\mathsf{F}}$, and to each $V \in \mathcal{V}_{\mathsf{S}}$ a context variable $X_V \in \mathcal{X}_{\mathsf{C}}$. The set of such first-order and context variables is denoted by $\mathcal{V}^{\mathcal{C}}$ and is called the *curried set of variables* corresponding to $\mathcal{V}$.

Definition 1. *The* currying function $\mathcal{C} : \mathcal{T}(\Sigma_{\mathsf{U}}, \mathcal{V}) \to \mathcal{T}(\Sigma_{\mathsf{U}}^{\mathcal{C}}, \mathcal{V}^{\mathcal{C}})$ *is defined recursively as follows:*

$$\begin{aligned} \mathcal{C}(f) &= a_f \\ \mathcal{C}(v) &= x_v \\ \mathcal{C}(f(r_1, \ldots, r_n, V)) &= X_V(\mathcal{C}(f(r_1, \ldots, r_n))) \\ \mathcal{C}(f(r_1, \ldots, r_n)) &= @(\mathcal{C}(f(r_1, \ldots, r_{n-1})), \mathcal{C}(r_n)), \text{ where } n > 0, \text{ and } r_n \notin \mathcal{V}_{\mathsf{S}}. \end{aligned}$$

We also define $\mathcal{C}(s \stackrel{?}{=} t)$ *as* $\mathcal{C}(s) \stackrel{?}{\approx} \mathcal{C}(t)$ *and extend it to unification problems.*

Using this definition we get $\mathcal{C}(f(b, V, f(v))) = @(X_V(@(a_f, a_b)), @(a_f, x_v))$.

The currying function can be extended to transform sequences of unranked terms into contexts. To do so, we extend Σ_U by a flexible arity function symbol $\diamond$, $\Sigma^\mathcal{C}_\mathsf{U}$ by the hole $\bullet$, define $\mathcal{C}(\diamond) = \bullet$, and then define the currying function for a sequence of unranked terms (that do not contain $\diamond$) as $\mathcal{C}(\langle r_1, \ldots, r_n\rangle) = \mathcal{C}(\diamond(r_1, \ldots, r_n))$. For instance, we get $\mathcal{C}(\langle V, a, f(a, b)\rangle) = \mathcal{C}(\diamond(V, a, f(a, b))) = @(@(X_V(\bullet), a_a), @(@(a_f, a_a), a_b))$.

We can use this extension to currify IS-substitutions to FC-substitutions: For an IS-substitution σ the corresponding $\mathcal{C}(\sigma)$ is defined as the substitution that maps each variable $\mathcal{C}(w)$ to $\mathcal{C}(\sigma(w))$ and any other variable to itself. For instance, currying $\sigma = [v \mapsto f(a, V), u \mapsto g(b), V \mapsto \langle U, a, b\rangle, U \mapsto V]$ gives $\mathcal{C}(\sigma) = [x_v \mapsto X_V(@(a_f, a_a)), x_u \mapsto @(a_g, a_b), X_V \mapsto @(@(X_U(\bullet), a_a), a_b), X_U \mapsto X_V(\bullet)]$ (assuming $X_U \neq X_V$).

Remark 1. It is interesting to notice that currying sequences of unranked terms produces contexts. Moreover the "shape" of these contexts will play a crucial role to prove the final result. In fact, the instantiations of context variables that we will consider must correspond to "curry forms" of sequences. This fact will allow us to prove that minimal solutions of the context equations resulting from curryfication process are rab and Strahler-bounded (see Lemmas 6 and 7 in next section), and prove that context unification restricted to this kind of unifiers is decidable.

Definition 2. *Given a term* $t \in \mathcal{T}(\Sigma^\mathcal{C}_\mathsf{U}, \mathcal{V}^\mathcal{C})$*, we say that it is* well-typed *(w.r.t.* Σ_U*), if* $\mathcal{C}^{-1}(t)$ *is defined, i.e. if there exists an* $r \in \mathcal{T}(\Sigma_\mathsf{U}, \mathcal{V})$ *such that* $\mathcal{C}(r) = t$*.*

Given a context $C \in \mathcal{T}(\Sigma^\mathcal{C}_\mathsf{U} \cup \{\bullet\}, \mathcal{V}^\mathcal{C})$*, we say that it is* well-typed *(w.r.t.* Σ_U*), if* $\mathcal{C}^{-1}(C)$ *is defined, i.e. if there exists a sequence* $\langle r_1, \ldots, r_n\rangle$*,* $r_i \in \mathcal{T}(\Sigma_\mathsf{U}, \mathcal{V})$*,* $1 \leq i \leq n$*, such that* $\mathcal{C}(\langle r_1, \ldots, r_n\rangle) = C$*.*

Let φ *be an FC-substitution such that* $\varphi(z) \in \mathcal{T}(\Sigma^\mathcal{C}_\mathsf{U} \cup \{\bullet\}, \mathcal{V}^\mathcal{C})$ *for all* $z \in \mathcal{V}^\mathcal{C}$*. We say that* φ *is* well-typed *(w.r.t.* Σ_U*), if* $\varphi(z)$ *is well-typed for all* $z \in \mathcal{V}^\mathcal{C}$*.*

Lemma 1. *For any IS-substitution* σ *and for any unranked term (or sequence of unranked terms)* r *over* Σ_U *and* $\mathcal{V}$*, we have* $\mathcal{C}(\sigma)(\mathcal{C}(r)) = \mathcal{C}(\sigma(r))$*.*

Proof: By structural induction on r. ■

Lemma 2. *If the sequence unification problem* Γ *over* Σ_U *and* $\mathcal{V}$ *is solvable, then the context unification problem* $\mathcal{C}(\Gamma)$ *over* $\Sigma^\mathcal{C}_\mathsf{U} \cup \{\bullet\}$ *and* $\mathcal{V}^\mathcal{C}$ *is also solvable.*

Proof: Let σ be a unifier of Γ. Then, by Lemma 1, it is easy to prove that $\mathcal{C}(\sigma)$ is a unifier of $\mathcal{C}(\Gamma)$. ■

In fact with the previous lemmas we have proved a stronger result: given a unifier σ of $l \stackrel{?}{=} r$, we can find a unifier $\mathcal{C}(\sigma)$ of $\mathcal{C}(l \stackrel{?}{=} r)$ that satisfies the property

$\mathcal{C}(\sigma(l)) = \mathcal{C}(\sigma)(\mathcal{C}(l))$. This property is represented by the commutativity of the following diagram:

$$\begin{array}{ccc} l \stackrel{?}{=} r & \xrightarrow{\mathcal{C}} & \mathcal{C}(l \stackrel{?}{=} r) \\ \Big\downarrow \sigma & \overset{\mathcal{C}}{\Longrightarrow} & \Big\downarrow \mathcal{C}(\sigma) \\ \sigma(l) & \xrightarrow{\mathcal{C}} & \mathcal{C}(\sigma)(\mathcal{C}(l)) \end{array}$$

Unfortunately, although we can currify a unifier of a sequence unification problem Γ to obtain a unifier of the context unification problem $\mathcal{C}(\Gamma)$, the converse is not true: $f(V) \stackrel{?}{=} g(f)$ is trivially unsolvable, but its curry form, $X_V(a_f) \stackrel{?}{\approx} @(a_g, a_f)$, is solvable: the substitution $[X_V \mapsto @(a_g, \bullet)]$ solves the context equation but there is no unifier for $f(V) \stackrel{?}{=} g(f)$. In general, solvability is not preserved by currying, i.e. the currying function is injective, but not surjective.

Example 1. The sequence unification problem

$$f(V_1, V_2) \stackrel{?}{=} f(f(a, V_2), f(V_2, a), b)$$

has these two unifiers:

$$\begin{array}{ll} \sigma_1 = \{V_1 \mapsto \langle f(a), f(a), b\rangle, & V_2 \mapsto \langle\,\rangle\} \\ \sigma_2 = \{V_1 \mapsto \langle f(a, b), f(b, a)\rangle, & V_2 \mapsto \langle b\rangle\} \end{array}$$

When currying the problem we get the context unification problem:

$$X_{V_2}(X_{V_1}(a_f)) \stackrel{?}{\approx} @(@(@(a_f, X_{V_2}(@(a_f, a_a))), @(X_{V_2}(a_f), a_a)), a_b)$$

that has the following four solutions:

$$\begin{array}{ll} \varphi_1 = \{X_{V_1} \mapsto @(@(@(\bullet, @(a_f, a_a)), @(a_f, a_a)), a_b), & X_{V_2} \mapsto \bullet\} \\ \varphi_2 = \{X_{V_1} \mapsto @(@(\bullet, @(@(a_f, a_a), a_b)), @(@(a_f, a_b), a_a)), & X_{V_2} \mapsto @(\bullet, a_b)\} \\ \varphi_3 = \{X_{V_1} \mapsto @(@(@(a_f, @(\bullet, a_a)), @(a_f, a_a)), a_b), & X_{V_2} \mapsto \bullet\} \\ \varphi_4 = \{X_{V_1} \mapsto @(@(@(a_f, @(a_f, a_a)), @(\bullet, a_a)), a_b), & X_{V_2} \mapsto \bullet\} \end{array}$$

It is easy to see that solutions φ_1 and φ_2 correspond respectively to σ_1 and σ_2: $\varphi_1 = \mathcal{C}(\sigma_1)$ and $\varphi_2 = \mathcal{C}(\sigma_2)$, while φ_3 and φ_4 do not have any such "corresponding" solutions.

In the previous example, substitution for variable X_{V_1} in solutions φ_3 and φ_4 are not "well-typed", i.e. they are not the curry form of any sequence of unranked terms. In φ_1 the variable X_{V_1} is mapped to the context $@(@(@(\bullet, @(a_f, a_a)), @(a_f, a_a)), a_b)$ that is the curry form of the sequence $\langle f(a), f(a), b\rangle$, whereas in φ_3 the variable X_{V_1} is mapped to the context $@(@(@(a_f, @(\bullet, a_a)), @(a_f, a_a)), a_b)$, that would be the curry form of something like $f(\langle a\rangle, f(a), b)$ which is not a sequence. In fact, $\mathcal{C}^{-1}$ is not defined

for $@(@(@(a_f, @(\bullet, a_a)), @(a_f, a_a)), a_b)$. Thus, we can not assert that we can always reconstruct a unifier for the original problem from the unifier that we get for its curry form, we will need these unifiers to be well-typed.

Slightly abusing the notation, for a well-typed FC-substitution φ we denote by $\mathcal{C}^{-1}(\varphi)$ the IS-substitution defined as follows: $(\mathcal{C}^{-1}(\varphi))(w) = \mathcal{C}^{-1}(\varphi(\mathcal{C}(w)))$, for each $w \in \mathcal{V}$.

Lemma 3. *Let Γ be a sequence unification problem over Σ_{U} and $\mathcal{V}$, and let $\mathcal{C}(\Gamma)$ be its curried form. Assume φ is a well-typed (w.r.t Σ_{U}) unifier of $\mathcal{C}(\Gamma)$, then $\mathcal{C}^{-1}(\varphi)$ is a unifier of Γ.*

Proof: Let $\mathcal{C}(l) \stackrel{?}{\approx} \mathcal{C}(r) \in \mathcal{C}(\Gamma)$. Then $\varphi(\mathcal{C}(l)) = \varphi(\mathcal{C}(r))$. Since φ, $\mathcal{C}(l)$, and $\mathcal{C}(r)$ are well-typed, we get that $\varphi(\mathcal{C}(l))$ and $\varphi(\mathcal{C}(r))$ are well-typed as well. Therefore, $\mathcal{C}^{-1}(\varphi(\mathcal{C}(l)))$ and $\mathcal{C}^{-1}(\varphi(\mathcal{C}(r)))$ exist and $\mathcal{C}^{-1}(\varphi(\mathcal{C}(l))) = \mathcal{C}^{-1}(\varphi(\mathcal{C}(r)))$. From this, by definition of $\mathcal{C}^{-1}$ for FC-substitutions we obtain $(\mathcal{C}^{-1}(\varphi))(l) = (\mathcal{C}^{-1}(\varphi))(r)$, i.e., $\mathcal{C}^{-1}(\varphi)$ is a unifier of $l \stackrel{?}{=} r \in \Gamma$. ■

Thus to preserve the set of solutions and ensure soundness in our transformation, i.e, to make the diagram commute, we can only consider well-typed unifiers. Now, we want to characterize these unifiers. As we have already argued in Remark 1, to be able to obtain a sequence from a context with $\mathcal{C}^{-1}$, the contexts must have a certain "shape".

Definition 3. *A* left-hole *context is a context that has the hole in its leftmost position, i.e. that can be built with this grammar:*

$$L \ ::= \ \bullet \mid X(\bullet) \mid @(L, t)$$

for context variable X and hole-free ranked term t.

Lemma 4. *Let φ be a ground FC-substitution such that $\varphi(z) \in \mathcal{T}(\Sigma_{\mathsf{U}}^{\mathcal{C}} \cup \{\bullet\}, \emptyset)$, for all $z \in \mathcal{V}^{\mathcal{C}}$. Then, φ is well-typed, iff $\varphi(X)$ is a left-hole context, for all context variable $X \in \mathcal{V}^{\mathcal{C}}$.*

Proof: By structural induction, from Definitions 2 and 1. ■

Now we define a variant of Context Unification, called Left-Hole Context Unification, as follows:

Definition 4. Left-Hole Context Unification *(*LHCU*) is a variant of Context Unification that requires instances of context variables to be left-hole contexts.*

Theorem 1. *Sequence Unification is P-reducible to Left-Hole Context Unification.*

Proof: The proof follows from Lemmas 2, 3 and 4. The $\mathcal{C}$ function is polynomial in the sum of the sizes of the terms of the equations. ■

Hence, currying preserves solvability: Γ is a solvable SU problem, iff $\mathcal{C}(\Gamma)$ is a solvable LHCU problem. Moreover, from each unifier of a sequence unification problem we can reconstruct a unifier of the corresponding left-hole context unification problem, and from each *ground* left-hole context unifier we can get a unifier of the original sequence unification problem. Notice that some *non-ground* left-hole context substitutions, like $[X \mapsto @(\bullet, @(y, a))]$, are not well-typed. The currying function is not onto, hence there are LHCU problems that are not the translation of any SU unification problem (see Section 5). Notice also that we assume that a LHCU problem is solvable, iff it has a *ground* left-hole context unifier. In fact, this is true, if we assume that the signature Σ_{R} contains at least a constant symbol.

4 Left-Hole Context Unification Decidability

In this section we reduce LHCU to Word Unification (WU) with Regular constraints, which is decidable [30]. Therefore, this reduction proves decidability of LHCU. The reduction is based on some ideas from [20]. There, it is proved (see Corollary 21) that if the rank-bound conjecture is true, then CU is decidable. The conjecture (see Conjecture 15) claims that there exists a computable upper bound for the Strahler number of some unifier of every solvable CU problem. Like in [20], the reduction will be done via the traversals of the terms that allows us to encode LHCU equations into WU equations. We need the regular constraints to make this encoding sound and ensure that the solutions of the WU equations really encode solutions of the corresponding LHCU problem. Here, we prove that there exists an upper bound for the rab and the Strahler numbers of minimal left-hole unifiers (see Lemmas 6 and 7).

In [21] it is proved that context unification is reducible to context unification with constants and only one binary symbol. The same reduction applies to LHCU. Therefore, from now on, we will assume that Σ_{R} only contains constants and a binary symbol that we represent as @. We also assume that Σ_{R} contains at least one constant. This is necessary to ensure that any solvable LHCU problem has a ground unifier. Moreover, we will also assume w.l.o.g. that we have just one initial context equation.

A naive encoding of a LHCU equation like $X(@(a, b)) \stackrel{?}{\approx} @(a, X(b))$ into a WU equation could be done using a postorder traversal of the terms of the equation as follows[2]:

$$\alpha_a\, \alpha_b\, \alpha_@\, W_X \stackrel{?}{=}_w \alpha_a\, \alpha_b\, W_X\, \alpha_@$$

where α_a, α_b and $\alpha_@$ are letters corresponding to a, b and @ respectively and W_X is the word variable that encodes the postorder traversal of the instantiation of the context variable X.

Then, some of the word solutions are:

$$\varphi_1 = [W_X \mapsto \epsilon]$$
$$\varphi_2 = [W_X \mapsto \alpha_@]$$

[2] We use $\stackrel{?}{=}_w$ to denote word equations.

where ϵ is the empty word. Notice that $\varphi_2(W_X)$ does not correspond to a postorder traversal of a context, while $\varphi_1(W_X)$ is the postorder traversal of the empty context $\bullet$. This forces us to impose regular constraints to this encoding. In what follows we will show that with regular constraints we can get a sound encoding.

Definition 5. *Given a LHCU problem Δ, we say that φ is a* minimal unifier*, if there exists a most general unifier ρ such that $\varphi = [x_1 \mapsto a, \ldots, x_n \mapsto a, X_1 \mapsto \bullet, \ldots, X_m \mapsto \bullet] \circ \rho$, where $\{x_1, \ldots, x_n, X_1, \ldots, X_m\} = vars(\rho(\Delta))$ and a is a constant of Σ_{R}.*

Notice that any solvable LHCU problem has a minimal unifier. The following is an adaptation of the sound and complete set of rules for Linear Second-Order Unification [18] to LHCU. Its soundness and completeness proof can be adapted from [18].

Definition 6. *The unification procedure is described by a set of problem transformations, where every transformation has the form*

$$\langle \Delta \cup \{s \stackrel{?}{\approx} t\}, \varphi\rangle \Longrightarrow \langle \rho(\Delta \cup \Delta'), \rho \circ \varphi\rangle$$

and is characterized by a rule $s \stackrel{?}{\approx} t \Longrightarrow \Delta'$ and a substitution ρ.

Simplification: $a \stackrel{?}{\approx} a \Longrightarrow \emptyset$,
$@(s_1, s_2) \stackrel{?}{\approx} @(t_1, t_2) \Longrightarrow \{s_1 \stackrel{?}{\approx} t_1, s_2 \stackrel{?}{\approx} t_2\}$,
$x \stackrel{?}{\approx} x \Longrightarrow \emptyset$, *and*
$X(s) \stackrel{?}{\approx} X(t) \Longrightarrow \{s \stackrel{?}{\approx} t\}$, *where $\rho = [\,]$ in the four cases.*

Projection: $X(s) \stackrel{?}{\approx} t \Longrightarrow \{s \stackrel{?}{\approx} t\}$ *and* $\rho = [X \mapsto \bullet]$.

Imitation: $X(s) \stackrel{?}{\approx} @(t_1, t_2) \Longrightarrow \{X'(s) \stackrel{?}{\approx} t_1\}$ *and* $\rho = [X \mapsto @(X'(\bullet), t_2)]$, *provided that X does not occur in t_2,*[3] *and*
$x \stackrel{?}{\approx} s \Longrightarrow \emptyset$ *and* $\rho = [x \mapsto s]$, *provided x does not occur in s.*

Flex-Flex: $X(s) \stackrel{?}{\approx} Y(t) \Longrightarrow \{X'(s) \stackrel{?}{\approx} t\}$ *and* $\rho = [X \mapsto Y(X'(\bullet))]$, *where $X \neq Y$.*

The transformations are applied starting with $\langle \Delta, [\,]\rangle$ until we get a pair of the form $\langle \emptyset, \varphi\rangle$, or no transformation is applicable. In the first case, φ is a unifier of Δ, and, in the second case, the problem is unsolvable.

Proposition 1. *The unification procedure described in Definition 6 is sound: if $\langle \Delta, [\,]\rangle \Longrightarrow^* \langle \emptyset, \varphi\rangle$, then φ is a unifier of Δ, and complete: if φ is a most general unifier of Δ, then there exists a transformation sequence of the form $\langle \Delta, [\,]\rangle \Longrightarrow^* \langle \emptyset, \varphi\rangle$.*[4]

[3] The violation of these provisos leads to an occur-check error in the equations.

[4] Notice that, for completeness, unifiers are required to be most general, but, in the soundness part, we can get non-most general unifiers.

Lemma 5. *Given a LHCU equation $s \stackrel{?}{\approx} t$, for any minimal unifier φ, and any context variable $X \in vars(s \stackrel{?}{\approx} t)$, we have $\varphi(X) = @(\ldots @(\bullet, \varphi(t_n)) \ldots, \varphi(t_1))$, where t_i is a subterm of s or of t, occurring as a second argument of an $@$, for all $1 \leq i \leq n$.*

Proof: From inspection of the transformation rules of Definition 6, we can see that right subterms (second arguments of $@$) are preserved: If $\langle \Delta_1, \varphi \rangle \Longrightarrow^* \langle \Delta_2, \rho \circ \varphi \rangle$ and t is a subterm of Δ_2 occurring as a second argument of an $@$, then there exists a subterm t' in Δ_1 such that t' also occurs as a second argument of an $@$, and $t = \rho(t')$.

Now, by inspection of the transformation rules we can also see that the only transformation rule that introduces new right-subterms is the imitation rule, that introduces a right-subterm of the equations as a new right-subterm in the substitution. Therefore, if $\langle \Delta_1, \varphi \rangle \Longrightarrow^* \langle \Delta_2, \rho \circ \varphi \rangle$, and t is a right-subterm of $\rho \circ \varphi(X)$, for some context variable X, then we can find an imitation step in the transformation sequence of one of the following forms:

$$\begin{array}{l} \langle \Delta_1, \varphi \rangle \Longrightarrow^* \langle \Delta', \rho' \circ \varphi \rangle \Longrightarrow \langle \Delta'', [Y \mapsto @(Y'(\bullet), s_2)] \circ \rho' \circ \varphi' \rangle \\ \qquad\qquad \Longrightarrow^* \langle \Delta_2, \rho'' \circ [Y \mapsto @(Y'(\bullet), s_2)] \circ \rho' \circ \varphi' \rangle \\ \langle \Delta_1, \varphi \rangle \Longrightarrow^* \langle \Delta', \rho' \circ \varphi \rangle \Longrightarrow \langle \Delta'', [y \mapsto s] \circ \rho' \circ \varphi' \rangle \\ \qquad\qquad \Longrightarrow^* \langle \Delta_2, \rho'' \circ [y \mapsto s] \circ \rho' \circ \varphi' \rangle \end{array}$$

where either $X = Y$, $t = \rho''(s_2)$ and s_2 is a right-subterm of Δ'; or Y [or y] is instantiated after X, and there is a right-subterm t' in s_2 [or s] such that $t = \rho''(t')$. Now, as we have proved, $s_2 = \rho'(t'')$ [or $t' = \rho'(t'')$], for some right-subterm t'' of Δ_1. Therefore, $t = \rho(t'')$, for some right-subterm t'' of Δ_1.

Completeness of the transformations ensure that any right-subterm of a most general unifier is an instance of a right-subterm of the original problem.

Finally, minimal unifiers can be obtained from most general unifiers, ensuring that instances of context variables have the form stated in the lemma. ■

The previous lemma allows us to prove that, if φ is a minimal unifier of $s \stackrel{?}{\approx} t$, then the number of times that we can go to the right descending through any branch of $\varphi(s)$, viewing the term as a tree, is bounded on the number of subterms of $s \stackrel{?}{\approx} t$.

Definition 7. *The* number of right accumulated branches (rab) *of a ground term $t \in \mathcal{T}^{\mathcal{C}}$, noted* $\mathrm{rab}(t)$*, is defined as:*

$$\begin{array}{ll} \mathrm{rab}(a) & = 0 \\ \mathrm{rab}(@(t_1, t_2)) & = \max\{\mathrm{rab}(t_1), 1 + \mathrm{rab}(t_2)\} \end{array}$$

Lemma 6. *Let $s \stackrel{?}{\approx} t$ be a LHCU equation and φ a minimal unifier, then* $\mathrm{rab}(\varphi(s)) \leq |s| + |t|$.

Proof: Lemma 5 ensures that, if φ is a minimal unifier, then for any subterm t_1 of $\varphi(s)$ occurring as a second-argument of an $@$ there exists a subterm t_2 in $s \stackrel{?}{\approx} t$ such that $t_1 = \varphi(t_2)$. Therefore, since there are $|s| + |t|$ subterms in $s \stackrel{?}{\approx} t$, and we can not repeat the same subterm in a branch, $\mathrm{rab}(\varphi(s)) \leq |s| + |t|$. ■

The definition of rab is similar to the definition of the Strahler number of a term:

Definition 8. *The* Strahler Number *of a term t built up from binary and nullary symbols, noted* $\mathrm{Strahler}(t)$*, is defined recursively as follows:*

$$\begin{aligned}\mathrm{Strahler}(a) &= 0\\ \mathrm{Strahler}(@(t_1,t_2)) &= \begin{cases}\mathrm{Strahler}(t_1)+1 & \text{if } \mathrm{Strahler}(t_1)=\mathrm{Strahler}(t_2)\\ \max\{\mathrm{Strahler}(t_1),\mathrm{Strahler}(t_2)\} & \text{otherwise}\end{cases}\end{aligned}$$

for any constant a, and the binary symbol $@$.

Since, we have $\mathrm{Strahler}(t) \leq \mathrm{rab}(t)$, for any term t, we can prove the following.

Lemma 7. *Given an LHCU equation $s \stackrel{?}{\approx} t$, all minimal unifiers φ satisfy* $\mathrm{Strahler}(\varphi(t)) \leq |s| + |t|$.

The previous lemma proves the rank-bound conjecture of [20] for a variant of context unification. Therefore, we can conclude decidability of LHCU from a small modification of the results of that paper. That proof was based on the use of traversals of terms, and on *traversal equations*. These traversal equations were reduced to word equations with regular constraints. Here, we find an easier way to constraint traversals of $\varphi(s)$ with regular expressions. These regular expressions define postorder traversals of terms with a bounded rab and allows us to avoid the use of traversal equations which can be replaced by simple word equations with regular constraints. What follows is then an alternative proof for the decidability of LHCU based on some ideas of [20].

Definition 9. *Let Σ_0 be the set containing a distinct letter α_a, for every constant $a \in \Sigma_{\mathsf{R}}$, and let $\alpha_@$ be also a letter (corresponding to the function symbol $@$).*[5] *Let us consider the following family of regular languages*

$$\begin{aligned}L^0 &= \Sigma_0\\ L^1 &= \Sigma_0\,(L^0\,\alpha_@)^*\\ &\cdots\\ L^n &= \Sigma_0\,(L^{n-1}\,\alpha_@)^*\end{aligned}$$

Lemma 8. *The language L^n defines the set of postorder traversals of ground terms $t \in \mathcal{T}(\Sigma_{\mathsf{R}}, \emptyset)$ satisfying* $\mathrm{rab}(t) \leq n$.

Theorem 2. *LHCU can be reduced to WU with regular constraints.*

Proof: Assume that, apart from $\alpha_@$ and from a distinct letter α_a, for every function symbol $a \in \Sigma_{\mathsf{R}}$, we also have a distinct word variable W_z, for every context or first-order variable $z \in \mathcal{X}$. The reduction uses the following transformation:

$$\begin{aligned}\mathcal{R}(a) &= \alpha_a\\ \mathcal{R}(@(t_1,t_2)) &= \mathcal{R}(t_1)\,\mathcal{R}(t_2)\,\alpha_@\\ \mathcal{R}(x) &= W_x\\ \mathcal{R}(X(t)) &= \mathcal{R}(t)W_X\end{aligned}$$

[5] Notice that from previous assumptions $\Sigma_{\mathsf{R}} \neq \emptyset$.

The translation is extended to context equations as $\mathcal{R}(s \stackrel{?}{\approx} t) = \mathcal{R}(s) \stackrel{?}{=}_w \mathcal{R}(t)$. We have to add the following regular constraints $W_x \in L^n$, for every first-order variable x, and $\alpha_a\, W_X \in L^n$, for every context variable X, where a is any of the constants of Σ_{R}, and $n = |s| + |t|$.

Extending the translation to substitutions, like it is done in the Section 3, we can prove easily that the translation maps minimal context unifiers to word unifiers. To prove that word unifiers may be decoded into context unifiers we need to use the regular constraints and Lemma 8. ■

Corollary 1. *Left-Hole Context Unification is decidable.*

5 Back to the Beginning

Now we look back at where we started from: Sequence Unification. Decidability of LHCU proved in the previous section gives another decidability proof of SU. Looking at the proof closer, we notice that we prove something more: Decidability of unification for an extension of SU. This extension, denoted ESU, is obtained if we allow individual variables to occur in functional positions, and a term to be applied to a sequence of terms. This is motivated by the fact that LHCU problems may contain terms like, e.g., $@(x_v, a)$ that could be obtained if we had currying defined for $v(a)$. We do not go into formal details here because of space limitation. The following example can serve for illustrating ESU:

Example 2. Extended sequence unification problem $\{f(a, V) \stackrel{?}{=} v(a, b)\}$ has two mgu's: $\sigma_1 = \{V \mapsto b, v \mapsto f()\}$ and $\sigma_2 = \{V \mapsto \langle U, a, b\rangle, v \mapsto f(a, U)\}$. Applying σ_2 to $v(a, b)$ gives $f(a, U, a, b)$.

Decidability of ESU can be shown based on decidability of LHCU. The sequence unification procedure [16] can be easily adapted to obtain a minimal complete unification procedure for ESU.

Moreover, we can transfer some of the results on complexity of Context Matching [29] to Extended Sequence Matching (ESM). The counterparts of linear context matching and varity 2 context matching problems are linear ESM (LESM) and varity 2 ESM (V2ESM), respectively. Shared-linear context matching gives a fragment of ESM that we call *prefix-closed ESM* (PCESM). It can be characterized by the following property: If a sequence variable V occurs in the subterms $f_1(r_1, \ldots, r_n, V, \ldots)$ and $f_2(l_1, \ldots, l_m, V, \ldots)$, where $f_1, f_2 \in \Sigma_{\mathsf{U}} \cup \mathcal{V}_{\mathsf{l}}$, then $f_1 = f_2$, $n = m$, and $r_i = l_i$ for each $1 \leq i \leq n$. It means that prefixes of all occurrences of a sequence variable should be the same. Then we have the following theorem, that follows from the analogous results in [29] and the construction of curry function:

Theorem 3. *LESM and PCESM are in P. V2ESM is NP-complete.*

It is hard to characterize a fragment of Sequence Matching obtained by inverse currying from Stratified Context matching. There is no obvious pattern in the form of such sequence matching problems.

6 Conclusion

We study the relation between two generalizations of Word Unification: Sequence Unification and Context Unification. We introduce a transformation function to translate sequence unification problems into context unification problems over a signature with constants and a single binary function symbol. The transformation preserves solvability in one direction: from SU to CU. To preserve solvability in the other direction, we add a restriction on the form of solutions of context unification problems, obtaining the left-hole variant of CU. We prove that a sequence unification problem is solvable iff the corresponding left-hole context unification problem is solvable, and the unifiers can be reconstructed in both directions. Moreover, we prove that LHCU is decidable, reducing it to WU with regular constraints. This result gives a decidability proof for an extension of SU, and, in particular, a new proof of decidability of SU. Based on the transformation, we transfer some complexity results from context matching to sequence matching.

References

1. H. Boley. *A Tight, Practical Integration of Relations and Functions*, volume 1712 of *LNAI*. Springer, 1999.
2. B. Buchberger, A. Crăciun, T. Jebelean, L. Kovács, T. Kutsia, K. Nakagawa, F. Piroi, N. Popov, J. Robu, M. Rosenkranz, and W. Windsteiger. Theorema: Towards computer-aided mathematical theory exploration. *J. Applied Logic*, 4:470–504, 2006.
3. E. Chasseur and Y. Deville. Logic program schemas, constraints and semi-unification. In *Proc. LOPSTR'97*, volume 1463 of *LNCS*, pages 69–89. Springer, 1998.
4. J. Coelho and M. Florido. CLP(Flex): Constraint logic programming applied to XML processing. In *On the Move to Meaningful Internet Systems 2004: CoopIS, DOA, and ODBASE. Proc. of Confederated Int. Conferences*, volume 3291 of *LNCS*, pages 1098–1112. Springer, 2004.
5. J. Coelho and M. Florido. VeriFLog: A constraint logic programming approach to verification of website content. In *Advanced Web and Network Technologies, and Applications*, volume 3842 of *LNCS*, pages 148–156. Springer, 2006.
6. Common Logic Working Group. Common Logic Standard. http://philebus.tamu.edu/cl/, 2007.
7. V. Diekert. Makanin's algorithm. In *Algebraic aspects of combinatorics on words*, chapter 12, pages 342–390. Cambridge University Press, 2002.
8. M. R. Genesereth, C. Petrie, T. Hinrichs, A. Hondroulis, M. Kassoff, N. Love, and W. Mohsin. Knowledge Interchange Format, draft proposed American National Standard (dpANS). Technical Report NCITS.T2/98-004, 1998.
9. M. L. Ginsberg. The MVL theorem proving system. *SIGART Bull.*, 2(3):57–60, 1991.
10. M. Hamana. Term rewriting with sequences. In: Proc. of the First Int. *Theorema* Workshop. Technical report 97–20, RISC, Linz, Austria, 1997.
11. P. Hayes and C. Menzel. Semantics of Knowledge Interchange Format. http://reliant.teknowledge.com/IJCAI01/HayesMenzel-SKIF-IJCAI2001.pdf, 2001.

12. P. J. Hayes and C. Menzel. Simple common logic. In *W3C Workshop on Rule Languages for Interoperability*. W3C, 2005.
13. J. Jaffar. Minimal and complete word unification. *J. ACM*, 37(1):47–85, 1990.
14. A. Koller. Evaluating context unification for semantic underspecification. In *3rd ESSLLI Student Session (ESSLLI'98), August 17-28*, pages 188–199, 1998.
15. T. Kutsia. Unification with sequence variables and flexible arity symbols and its extension with pattern-terms. In *Proc. of Joint Artificial Intelligence, Automated Reasoning and Symbolic Computation, AISC'2002, Calculemus'02 Conference*, volume 2385 of *LNAI*, pages 290–304. Springer, 2002.
16. T. Kutsia. Solving equations with sequence variables and sequence functions. *Journal of Symbolic Computation*, 42(3):352–388, 2007.
17. T. Kutsia and M. Marin. Matching with regular constraints. In *Proc. of LPAR'05*, volume 3835 of *LNAI*, pages 215–229. Springer, 2005.
18. J. Levy. Linear second-order unification. In *Proceedings of the 7th International Conference on Rewriting Techniques and Applications (RTA'96)*, volume 1103 of *LNCS*, pages 332–346, 1996.
19. J. Levy, J. Niehren, and M. Villaret. Well-nested context unification. In *Proc. of the 20th Int. Conf. on Automated Deduction, CADE-20*, volume 3632 of *Lecture Notes in Artificial Intelligence*, pages 149–163. Springer-Verlag, 2005.
20. J. Levy and M. Villaret. Context unification and traversal equations. In *Proceedings of the 12th International Conference on Rewriting Techniques and Applications (RTA'01)*, volume 2041 of *LNCS*, pages 169–184, 2001.
21. J. Levy and M. Villaret. Currying second-order unification problems. In *Proceedings of the 13th International Conference on Rewriting Techniques and Applications (RTA'02)*, volume 2378 of *LNCS*, pages 326–339, 2002.
22. G. S. Makanin. The problem of solvability of equations in a free semigroup. *Math. USSR Sbornik*, 32(2):129–198, 1977.
23. M. Marin and T. Kutsia. Foundations of the rule-based system RhoLog. *Journal of Applied Non-Classical Logics*, 16(1–2):151–168, 2006.
24. J. Niehren, M. Pinkal, and P. Ruhrberg. A uniform approach to underspecification and parallelism. In *Proc. of the 35th Annual Meeting of the ACL and the 8th Conf. of the European Chapter of the ACL (ACL'97)*, pages 410–417, 1997.
25. L. Paulson. Isabelle: the next 700 theorem provers. In P. Odifreddi, editor, *Logic and Computer Science*, pages 361–386. Academic Press, 1990.
26. W. Plandowski. Satisfiability of word equations with constants is in PSPACE. In *Proc. of the 40th Annual Symp. on Foundations of Computer Science, FOCS'99*, pages 495–500, New York City, USA, 1999. IEEE Press.
27. J. Richardson and N. E. Fuchs. Development of correct transformation schemata for Prolog programs. In *Proc. of the 7th Int. Workshop on Logic Program Synthesis and Transformation*, volume 1463 of *LNCS*, pages 263–281. Springer, 1997.
28. M. Schmidt-Schauß. A decision algorithm for stratified context unification. *Journal of Logic and Computation*, 12:929–953, 2002.
29. M. Schmidt-Schauß and J. Stuber. The complexity of linear and stratified context matching problems. *Theory of Computing Syst.*, 37(6):717–740, 2004.
30. K. U. Schulz. Makanin's algorithm for word equations – two improvements and a generalization. In *Proc. of Word Equations and Related Topics (IWWERT'90)*, volume 572 of *LNCS*, pages 85–150, Tübingen, Germany, 1990. Springer.
31. S. Wolfram. *The Mathematica Book*. Wolfram Media, 5th edition, 2003.