On the Relation Between Context and Sequence Unification

Temur Kutsia

Research Institute for Symbolic Computation (RISC), Johannes Kepler University of Linz, Austria.

Jordi Levy

Artificial Intelligence Research Institute (IIIA), Spanish Council for Scientific Research (CSIC), Barcelona, Spain.

Mateu Villaret

Departament d'Informàtica i Matemàtica Aplicada (IMA), Universitat de Girona (UdG), Girona, Spain.

Abstract

Both Sequence and Context Unification generalize the same problem: Word Unification. Besides that, Sequence Unification solves equations between unranked terms involving sequence variables, and seems to be appealing for information extraction in XML documents, program transformation, knowledge representation, and rule-based programming. It is decidable. Context Unification deals with the same problem for ranked terms involving context variables, and has applications in computational linguistics and program transformation. Its decidability is a long-standing open question.

In this work we study a relation between these two problems. We introduce a variant (restriction) of Context Unification, called Left-Hole Context Unification (LHCU), to which Sequence Unification is P-reduced: We define a partial currying procedure to translate Sequence Unification problems into Left-Hole Context Unification problems, and prove the 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 new decidability proof for Sequence Unification. Finally, we define an extension of Sequence Unification (ESU) and, closing the circle, prove the inter-P-reducibility of LHCU and ESU.

Key words: Sequence Unification, Context Unification, Word Unification.

1. Introduction

In this work we study a relationship between Sequence and Context Unification. Both problems are generalizations of Word Unification (Makanin, 1977; Jaffar, 1990; Schulz, 1990; Plandowski, 1999; Diekert, 2002). Word Unification is the problem of solving equations between terms built up from letters and word variables (or unknowns). A solution of a word equation is a mapping from variables to words that when applied to both sides of the equation gives the same word.

Sequence Unification (SU) is the problem of solving equations between terms built up using an unranked signature (aka flexible arity, or variadic function symbols) 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 (Kutsia, 2002, 2007).

Solving equations with sequence variables has quite a broad range of applications. The rule-based programming language of Mathematica (Wolfram, 2003) 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 (Richardson and Fuchs, 1997; Chasseur and Deville, 1998) used in synthesis and transformation of logic programs. Other applications include knowledge representation (Genesereth et al., 1998; Hayes and Menzel, 2001, 2005), automated reasoning (Paulson, 1990; Ginsberg, 1991; Buchberger et al., 2006), rewriting (Hamana, 1997), functional logic programming (Boley, 1999). The ISO standard proposal for Common Logic (Common Logic Working Group, 2007) has notation for sequence variables (called there sequence markers). Recently there have been developments (Coelho and Florido, 2004, 2006) in XML querying and transformation that model XML documents with terms over an unranked signature and use sequence matching and unification techniques (Kutsia, 2002) for querying, transforming, and verifying them. Sequence equation and disequation solving was used in collaborative development of XML schema (Coelho et al., 2007). Obviously, we can not give an exhaustive overview of all the applications here.

Context Unification (CU) is the problem of solving equations between terms built up using a ranked signature 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 (Levy and Villaret, 2002). Nevertheless several fragments and variants are known to be decidable (Levy, 1996; Schmidt-Schauß, 2002; Levy et al., 2005). The main application field of Context Unification is computational linguistics, mainly in compositional semantics of natural language (Niehren et al., 1997; Koller, 1998; Niehren and Villaret, 2002, 2005; Levy et al., 2005). Matching algorithms for multi-hole

⋆ A preliminary version of this paper (Kutsia et al. (2007)) was presented at the 18th Int. Conf. on Rewriting Techniques and Applications, RTA'07.

Email addresses: kutsia@risc.uni-linz.ac.at (Temur Kutsia), levy@iiia.csic.es (Jordi Levy), villaret@ima.udg.edu (Mateu Villaret).

URLs: http://www.risc.uni-linz.ac.at/people/tkutsia/ (Temur Kutsia), http://www.iiia.csic.es/~levy (Jordi Levy), http://ima.udg.es/~villaret (Mateu Villaret).

contexts have been designed with the goal to use them in program transformation (Chiba et al., 2005) and XML processing (Okui and Suzuki, 2006).

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 (Kutsia and Marin, 2005; Marin and Kutsia, 2006).

The goal of this paper is to look in depth into relations between Sequence and Context Unification. We define a curryfication transformation that translates Sequence Unification problems into Context Unification problems over a signature consisting of constants and a single binary function symbol @ (curried context unification problems). The transformation "encodes" sequence variables into context variables while individual variables become first-order variables. It 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 (LHCU).* Theorem 17 proves that Sequence Unification is P-reducible to Left-Hole Context Unification. Later, we prove that Left-Hole Context Unification is a decidable variant of Context Unification. We do it by reducing (see Theorem 33) Left-Hole Context Unification to *Word Unification with Regular Constraints (WURC)* that is known to be decidable (Schulz, 1990). The reduction transforms context equations into word equations on the postorder traversal of the terms. Regular constraints are required to filter the solutions of the word equations that correspond to traversals of terms. This reduction is based on some ideas of (Levy and Villaret, 2001).

With these reductions we get a new decidability proof for Sequence Unification (SU), shorter than the one in (Kutsia, 2007). In addition we also get decidability for an extension of Sequence Unification. In fact, we prove that the *Extended Sequence Unification (ESU)* and the Left-Hole Context Unification(LHCU) problems are mutually P-reducible (see Corollary 40). Moreover, this translation also allows us to use certain complexity results for context matching of (Schmidt-Schauß and Stuber, 2004) to characterize complexity of fragments of sequence matching. All these P-reductions are represented in the following diagram:

$$\begin{tikzcd}
SU \arrow[r, "Th.\ 17"] \arrow[d, hook] & LHCU \arrow[r, "Th.\ 33"] \arrow[dl, leftrightarrow, "Cor.\ 40"] & WURC \\
ESU & &
\end{tikzcd}$$

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 proved. Section 5 describes an extension of Sequence Unification thanks to the currying process, shows some complexity results for Extended Sequence Matching, and describes a unification procedure. Section 6 is the conclusion.

2. Preliminary Definitions

2.1. Sequence Unification

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:

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

and sequences of unranked terms by

$$seq ::= \langle r_1, \ldots, r_n \rangle$$

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. For the sake of readability, we sometimes put sequences between angular brackets, that are later flattened: for instance, the replacement of V by $\langle f(b), c\rangle$ in $f(u, V)$ results in $f(a, \langle f(b), c\rangle) = f(a, f(b), c)$. We do not distinguish between a singleton sequence and its sole member. [1] The length of the sequence $\langle r_1, \ldots, r_n\rangle$, $n \geq 0$, is n.

The set of unranked terms over Σ_{U} and $\mathcal{V}$ is denoted by $\mathcal{T}(\Sigma_{\mathsf{U}}, \mathcal{V})$. The letters f, g, a, b and c will be used for function symbols, v and u for individual variables, V and U for sequence variables, w for individual or sequence variables, r and l for unranked terms, and $\tilde{r}$ for sequences of unranked terms. We call unranked terms from $\mathcal{T}(\Sigma_{\mathsf{U}}, \mathcal{V}) \setminus \mathcal{V}_{\mathsf{S}}$ the *individual terms.*

The *size* of a term r is the measure of its number of symbols and is denoted by $|r|$. We denote by $vars(r)$ the *set of variables* of a term. These definitions are generalized for any syntactic object throughout the paper.

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. We use the Greek letters σ and ϑ to denote IS-substitutions.

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 $\sigma(V) \neq V$. We say that $dom(\sigma) = \{v_1, \ldots, v_n, V_1, \ldots, V_m\}$ is the *domain* of σ.

The domain of IS-substitutions can be extended from the sets of variables to sets of unranked terms using the congruence

$$\sigma(f(r_1, \ldots, r_n)) = f(\sigma(r_1), \ldots, \sigma(r_n)).$$

Similarly, σ can be applied to a sequence of unranked terms using

$$\sigma(\langle r_1, \ldots, r_n\rangle) = \langle \sigma(r_1), \ldots, \sigma(r_n)\rangle.$$

[1] Although this could seem a source of possible error, it works in our setting. For instance, later, when we currify unranked terms, we use two functions $\mathcal{C}$ or $\overline{\mathcal{C}}$ (depending on the context) to currify terms, or sequences of terms, respectively. The fact that we do not distinguish between a sequence of a sole term and one term does not lead to errors. Notice also that, in our setting, sequences do not occur in unranked terms, only in substitutions.

We call $\sigma(r)$ (respectively $\sigma(\langle r_1,\dots,r_n\rangle)$) an *instance* of r (respectively of $\langle r_1,\dots,r_n\rangle$) 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.

The *restriction* of a substitution ϑ to a set of variables W, denoted $\vartheta|_W$, is a substitution defined as follows: $\vartheta|_W(w) = \vartheta(w)$ if $w \in W$, and $\vartheta|_W(w) = w$ otherwise.

The *composition* of two IS-substitutions σ and ϑ, written as $\sigma \circ \vartheta$, is defined by $(\sigma \circ \vartheta)(r) = \sigma(\vartheta(r))$. 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 $w \in W$.

Definition 1. A *sequence unification problem* (SU problem) is a finite set of *equations* (unoriented pairs) of individual terms, denoted $\{l_1 \stackrel{?}{=} r_1,\dots,l_n \stackrel{?}{=} r_n\}$.

IS-Substitutions extend to equations and unification problems: $\sigma(l_1 \stackrel{?}{=} r_1) = \sigma(l_1) \stackrel{?}{=} \sigma(r_1)$ and $\sigma(\{e_1,\dots,e_n\}) = \{\sigma(e_1),\dots,\sigma(e_n)\}$. 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. Context Unification

A ranked signature $\Sigma_{\mathsf{R}} = \bigcup_{i\geq 0} \Sigma_i$ is a finite set of fixed *arity* function symbols, where all symbols of Σ_i have arity equal to i. Additionally to Σ_{R} we also use the 0-ary symbol $\bullet$, called the *hole*. Given Σ_{R}, a countable set of first-order variables $\mathcal{X}_0$, and a countable set of context variables $\mathcal{X}_1$, we define *ranked terms over* $\Sigma_{\mathsf{R}} \cup \{\bullet\}$ *and* $\mathcal{X} = \mathcal{X}_0 \cup \mathcal{X}_1$ by the following grammar:

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

where $x \in \mathcal{X}_0$, $X \in \mathcal{X}_1$, and $f \in \Sigma_n$. When $n = 0$, we omit the parentheses and write just f. The sets Σ_{R}, $\mathcal{X}_0$ and $\mathcal{X}_1$ are mutually disjoint. *Constants* are 0-ary function symbols. The set of ranked terms over Σ_{R} and $\mathcal{X}$ is denoted by $\mathcal{T}(\Sigma_{\mathsf{R}},\mathcal{X})$.

Definition 2. A *context* is a ranked term with exactly one occurrence of the hole. *Application* of a context C to a ranked term t, written $C[t]$, 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.

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. Restriction and composition are 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 $\varphi(X) \neq X(\bullet)$. Domains are defined as above.

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

$$\begin{aligned}\varphi(X(s)) &= \varphi(X)[\varphi(s)],\\ \varphi(f(s_1, \ldots, s_n)) &= f(\varphi(s_1), \ldots, \varphi(s_n)).\end{aligned}$$

Definition 3. A *context unification problem* (CU problem) is a finite set of *equations* (unoriented pairs) of ranked *hole-free* terms, denoted $\{s_1 \overset{?}{\approx} t_1, \ldots, s_n \overset{?}{\approx} t_n\}$.

A *unifier* of a context unification problem Δ is an FC-substitution φ such that $\varphi(s) = \varphi(t)$ for each $s \overset{?}{\approx} t \in \Delta$, and Δ is solvable, if it has a unifier. The notions of a ground, 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 *curryfication* transformation that will serve us to transform sequence unification problems into (a variant of) context unification problems.

The currying function, defined below, transforms unranked terms into ranked terms, sequence equations (problems) into context equations (problems), sequences of unranked terms into contexts and IS-substitutions into FC-substitutions.

Definition 4. Given an unranked signature Σ_{U}, we define its ranked *curried signature* as $\Sigma_{\mathsf{U}}^{\mathcal{C}} = \Sigma_0 \cup \Sigma_2$, where Σ_0 contains a unique and distinct constant $a_f \neq \bullet$, for each $f \in \Sigma_{\mathsf{U}}$, $\Sigma_2 = \{@\}$, and $\Sigma_i = \emptyset$, for $i \neq 0, 2$.

Similarly, given the set of individual and sequence variables $\mathcal{V} = \mathcal{V}_{\mathsf{I}} \cup \mathcal{V}_{\mathsf{S}}$, we define its *curried set of variables* $\mathcal{V}^{\mathcal{C}} = \mathcal{X}_0 \cup \mathcal{X}_1$, where $\mathcal{X}_0$ contains a first-order variable x_v for each individual variable $v \in \mathcal{V}_{\mathsf{I}}$, and $\mathcal{X}_1$ a context variable X_V for each sequence variable $V \in \mathcal{V}_{\mathsf{S}}$.

We define the *currying functions* $\mathcal{C}$ over unranked terms and sequence equations; and $\overline{\mathcal{C}}$ over sequences of unranked terms, as follows:

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

$$\begin{aligned}\overline{\mathcal{C}}(\langle\rangle) &= \bullet,\\ \overline{\mathcal{C}}(\langle r_1, \ldots, r_n, V\rangle) &= X_V(\overline{\mathcal{C}}(\langle r_1, \ldots, r_n\rangle)),\\ \overline{\mathcal{C}}(\langle r_1, \ldots, r_n\rangle) &= @(\overline{\mathcal{C}}(\langle r_1, \ldots, r_{n-1}\rangle), \mathcal{C}(r_n)), \text{ where } n \geq 1 \text{ and } r_n \notin \mathcal{V}_{\mathsf{S}}\end{aligned}$$

with $f \in \Sigma_{\mathsf{U}}$, $v \in \mathcal{V}_{\mathsf{I}}$, $V \in \mathcal{V}_{\mathsf{S}}$, $r_1, \ldots, r_n \in \mathcal{T}(\Sigma_{\mathsf{U}}, \mathcal{V})$, and a_f, x_v, and X_V being the associated counterparts of f, v, and V, respectively.

We define the currying function $\mathcal{C}$ to transform IS-substitutions into FC-substitutions as follows: for every $v \in \mathcal{V}_\mathsf{I}$, $\mathcal{C}(\sigma)$ maps the first-order variable $\mathcal{C}(v)$ to $\mathcal{C}(\sigma(v))$, for every $V \in \mathcal{V}_\mathsf{S}$, $\mathcal{C}(\sigma)$ maps the context variable $\mathcal{C}(V)$ to $\overline{\mathcal{C}}(\sigma(V))$, and any other variable to itself.

Example 5. The curryfication of

$$\sigma = [v \mapsto f(a, V),\ U \mapsto f(a, u),\ W \mapsto \langle V, a, b\rangle,\ V \mapsto u]$$

results in

$$\begin{aligned}\mathcal{C}(\sigma) = {} & [x_v \mapsto X_V(@(a_f, a_a)),\ X_U \mapsto @(\bullet, @(@(a_f, a_a), x_u)),\\ & X_W \mapsto @(@(X_V(\bullet), a_a), a_b),\ X_V \mapsto @(\bullet, x_u)].\end{aligned}$$

Remark 6. Notice that $\mathcal{C}$ is not defined for non-singleton sequences, and for singleton sequences: $@(\bullet, a_f) = \overline{\mathcal{C}}(\langle f\rangle) = \overline{\mathcal{C}}(f) \neq \mathcal{C}(f) = a_f$.

It is interesting to notice that currying sequences of unranked terms with $\overline{\mathcal{C}}$ produces contexts. We have the following lemma:

Lemma 7. *The function $\overline{\mathcal{C}}$ for sequences and the function $\mathcal{C}$ for individual terms satisfy the following equalities:*

$$\begin{aligned}\mathcal{C}(f(r_1, \ldots, r_n, r'_1, \ldots, r'_m)) &= \overline{\mathcal{C}}(\langle r'_1, \ldots, r'_m\rangle)[\mathcal{C}(f(r_1, \ldots, r_n))].\\ \overline{\mathcal{C}}(\langle r_1, \ldots, r_n, r'_1, \ldots, r'_m\rangle) &= \overline{\mathcal{C}}(\langle r'_1, \ldots, r'_m\rangle)[\overline{\mathcal{C}}(\langle r_1, \ldots, r_n\rangle)].\end{aligned}$$

Proof. By induction on m. When $m = 0$, the equalities are straightforward. Assume they hold for $m = k$ and prove for $m = k + 1$. First, assume that $r'_{m+1} = V$ for some V. Then we have:

$$\begin{aligned}\mathcal{C}(f(r_1, \ldots, r_n, r'_1, \ldots, r'_m, V)) &= \text{(by the definition of } \mathcal{C})\\ X_V(\mathcal{C}(f(r_1, \ldots, r_n, r'_1, \ldots, r'_m)) &= \text{(by the induction hypothesis)}\\ X_V(\overline{\mathcal{C}}(\langle r'_1, \ldots, r'_m\rangle)[\mathcal{C}(f(r_1, \ldots, r_n))]) &= \text{(a property of context application)}\\ X_V(\overline{\mathcal{C}}(\langle r'_1, \ldots, r'_m\rangle))[\mathcal{C}(f(r_1, \ldots, r_n))] &= \text{(by the definition of } \overline{\mathcal{C}})\\ \overline{\mathcal{C}}(\langle r'_1, \ldots, r'_m, X_V\rangle)[\mathcal{C}(f(r_1, \ldots, r_n))].&\end{aligned}$$

Now, assume that $r'_{m+1} = r$ is an individual term. Then

$$\begin{aligned}\mathcal{C}(f(r_1, \ldots, r_n, r'_1, \ldots, r'_m, r)) &= \text{(by the definition of } \mathcal{C})\\ @(\mathcal{C}(f(r_1, \ldots, r_n, r'_1, \ldots, r'_m), \mathcal{C}(r)) &= \text{(by the induction hypothesis)}\\ @(\overline{\mathcal{C}}(\langle r'_1, \ldots, r'_m\rangle)[\mathcal{C}(f(r_1, \ldots, r_n))], \mathcal{C}(r)) &= \text{(since } \mathcal{C}(r) \text{ is not a context)}\\ @(\overline{\mathcal{C}}(\langle r'_1, \ldots, r'_m\rangle), \mathcal{C}(r))[\mathcal{C}(f(r_1, \ldots, r_n))] &= \text{(by the definition of } \overline{\mathcal{C}})\\ \overline{\mathcal{C}}(\langle r'_1, \ldots, r'_m, r\rangle)[\mathcal{C}(f(r_1, \ldots, r_n))].&\end{aligned}$$

The second equality can be proved analogously. □

In particular, we have $\mathcal{C}(f(r_1, \ldots, r_n)) = \overline{\mathcal{C}}(\langle r_1, \ldots, r_n\rangle)[a_f]$.

The "shape" of the contexts $\overline{\mathcal{C}}$ produces will play a crucial role to prove the final result. In particular, the hole always occurs in a left-most inner-most position. We will only consider instantiations of context variables that correspond to "curry forms" of sequences. This fact will allow us to prove that Context Unification restricted to this kind of unifiers is decidable.

It is easy to see that $\overline{\mathcal{C}}$ is injective, therefore $\overline{\mathcal{C}}^{-1}$ is uniquely defined. As we mention in the previous remark, the contexts produced by $\overline{\mathcal{C}}$ have a special form, what means that $\overline{\mathcal{C}}^{-1}(C)$ is not defined for all contexts $C \in \mathcal{T}(\Sigma_{\mathsf{U}}^{\mathcal{C}} \cup \{\bullet\}, \mathcal{V}^{\mathcal{C}})$. For instance, it is not defined for $@(a_a, X_V(\bullet))$. Moreover, $\mathcal{C}^{-1}(t)$ is not defined for all terms $t \in \mathcal{T}(\Sigma_{\mathsf{U}}^{\mathcal{C}}, \mathcal{V}^{\mathcal{C}})$. In particular, t cannot contain subterms of the form $X_V(x_u)$ or $@(x_u, t')$.[2] This motivates the following definition.

Definition 8. Given a ranked term $t \in \mathcal{T}(\Sigma_{\mathsf{U}}^{\mathcal{C}}, \mathcal{V}^{\mathcal{C}})$, we say that it is *legal* (w.r.t. Σ_{U} and $\mathcal{V}$), 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 $C \in \mathcal{T}(\Sigma_{\mathsf{U}}^{\mathcal{C}} \cup \{\bullet\}, \mathcal{V}^{\mathcal{C}})$, we say that it is *legal* (w.r.t. Σ_{U} and $\mathcal{V}$), if $\overline{\mathcal{C}}^{-1}(C)$ is defined, i.e. if there exists a sequence $\langle r_1, \dots, r_n\rangle$, with $r_i \in \mathcal{T}(\Sigma_{\mathsf{U}}, \mathcal{V})$, for $1 \le i \le n$, such that $\overline{\mathcal{C}}(\langle r_1, \dots, r_n\rangle) = C$.

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

Lemma 9. *For any IS-substitution σ and for any unranked term r and sequence of unranked terms $\tilde{r}$ over Σ_{U} and $\mathcal{V}$, we have $\mathcal{C}(\sigma)(\mathcal{C}(r)) = \mathcal{C}(\sigma(r))$ and $\mathcal{C}(\sigma)(\overline{\mathcal{C}}(\tilde{r})) = \overline{\mathcal{C}}(\sigma(\tilde{r}))$.*

Proof. By structural induction on r and $\tilde{r}$, using Lemma 7. $\square$

Lemma 10. *If the sequence unification problem Γ over Σ_{U} and $\mathcal{V}$ is solvable, then the context unification problem $\mathcal{C}(\Gamma)$ over $\Sigma_{\mathsf{U}}^{\mathcal{C}} \cup \{\bullet\}$ and $\mathcal{V}^{\mathcal{C}}$ is also solvable.*

Proof. Let σ be a unifier of Γ. Then, by Lemma 9, it is easy to prove that $\mathcal{C}(\sigma)$ is a unifier of $\mathcal{C}(\Gamma)$. $\square$

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

$$\begin{tikzcd}
l \overset{?}{=} r \arrow[r, "\mathcal{C}"] \arrow[dd] & \mathcal{C}(l \overset{?}{=} r) \arrow[dd] \\
\sigma \arrow[r, Rightarrow, "\mathcal{C}"] & \mathcal{C}(\sigma) \\
\sigma(l) \arrow[r, "\mathcal{C}"] & \mathcal{C}(\sigma)(\mathcal{C}(l))
\end{tikzcd}$$

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: $f(V) \overset{?}{=} g(f)$ is unsolvable, but its curry form, $X_V(a_f) \overset{?}{\approx} @(a_g, a_f)$, is solvable: the substitution $[X_V \mapsto @(a_g, \bullet)]$ solves it. In general, solvability is not preserved by currying, i.e. the currying function is injective, but not surjective.

[2] In Section 5 by extending the set of sequence terms and the domain of the curry function, it will become exhaustive.

Example 11. The sequence unification problem

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

has these two unifiers:

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

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:

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

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, substitutions for the variable X_{V_1} in solutions φ_3 and φ_4 are not legal, 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, $\overline{\mathcal{C}}^{-1}$ is not defined for $@(@(@(a_f, @(\bullet, a_a)), @(a_f, a_a)), a_b)$. Thus, we cannot assert that we can always reconstruct a unifier for the original problem from the unifier that we get for its curry form, unless the unifier of the curry form is legal.

Lemma 12. *For any legal FC-substitution φ, $\mathcal{C}^{-1}(\varphi)$ is defined and satisfies the equality $(\mathcal{C}^{-1}(\varphi))(v) = \mathcal{C}^{-1}(\varphi(\mathcal{C}(v)))$ for all $v \in \mathcal{V}_\mathsf{I}$, and $(\mathcal{C}^{-1}(\varphi))(V) = \overline{\mathcal{C}}^{-1}(\varphi(\overline{\mathcal{C}}(V)))$ for all $V \in \mathcal{V}_\mathsf{S}$.*

Proof. By definition of $\mathcal{C}$ for IS-substitutions and of legality. $\square$

Lemma 13. *Let Γ be a sequence unification problem over Σ_U and $\mathcal{V}$, and let $\mathcal{C}(\Gamma)$ be its curried form. Assume that φ is a legal (w.r.t Σ_U) unifier of $\mathcal{C}(\Gamma)$, then $\mathcal{C}^{-1}(\varphi)$ is a unifier of Γ.*

Proof. By structural induction, from $(\mathcal{C}^{-1}(\varphi))(w) = \mathcal{C}^{-1}(\varphi(\mathcal{C}(w)))$ (Lemma 12), for variables $w \in \mathcal{V}$, we can prove $(\mathcal{C}^{-1}(\varphi))(r) = \mathcal{C}^{-1}(\varphi(\mathcal{C}(r)))$ for any unranked term $r \in \mathcal{T}(\Sigma_\mathsf{U}, \mathcal{V})$.

Similarly, we can prove that, if the FC-substitution φ and the ranked term t are legal, then $\varphi(t)$ is also legal.

Let $\mathcal{C}(l) \overset{?}{\approx} \mathcal{C}(r) \in \mathcal{C}(\Gamma)$. Then $\varphi(\mathcal{C}(l)) = \varphi(\mathcal{C}(r))$. Since φ, $\mathcal{C}(l)$, and $\mathcal{C}(r)$ are legal, we get that $\varphi(\mathcal{C}(l))$ and $\varphi(\mathcal{C}(r))$ are legal 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, we obtain $(\mathcal{C}^{-1}(\varphi))(l) = (\mathcal{C}^{-1}(\varphi))(r)$, i.e., $\mathcal{C}^{-1}(\varphi)$ is a unifier of $l \overset{?}{=} r \in \Gamma$. $\square$

Thus, to preserve the set of solutions and to ensure soundness in our transformation, i.e, to make the diagram commute, we can only consider legal unifiers. Now, we want to characterize these unifiers. As we have already argued in Remark 6, to be able to obtain a sequence from a context with $\overline{\mathcal{C}}^{-1}$, the contexts must have a certain "shape".

Definition 14. A *left-hole* context is a context that has the hole in its left-most 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 15. *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 legal iff $\varphi(X)$ is a left-hole context for all context variables $X \in \mathcal{V}^{\mathcal{C}}$.*

Proof. By structural induction, from Definitions 4 and 8. Notice that ground and hole-free terms are always legal, therefore the legality requirement only applies to instances of context variables. $\square$

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

Definition 16. *Left-Hole Context Unification* (*LHCU*) is a variant of Context Unification that requires instances of context variables to be left-hole contexts.

Theorem 17. *Sequence Unification is P-reducible to Left-Hole Context Unification.*

Proof. The proof follows from Lemmas 10, 13 and 15. The $\mathcal{C}$ function is polynomial in the sum of the sizes of the terms of the equations. $\square$

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 legal. The currying function is not onto, hence there are LHCU problems that are not the translation of any SU unification problem (see Section 5). This makes the reduction of Left-Hole Context Unification to Sequence Unification impossible by means of curryfication.

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. If a LHCU problem has a non-ground unifier, and Σ_{R} contains a constant a, we can get a ground unifier instantiating every first-order variable by a, and every context variable by $@(\bullet, a)$.

4. Left-Hole Context Unification Decidability

In this section we reduce LHCU to Word Unification (WU) with Regular constraints, which is decidable (Schulz, 1990). Therefore, this reduction proves decidability of LHCU. The reduction is based on some ideas from (Levy and Villaret, 2001). There, it is proved (see Corollary 21) that if the rank-bound conjecture is true, then CU is decidable. The conjecture has never been proved, and the decidability of CU remains as an open question. Here (see Lemmas 23 and 25) we prove a stronger result for left-hole unifiers, which leads us to prove decidability of LHCU. Like in (Levy and Villaret, 2001), the reduction will be done via the traversals of the terms that allows us to encode LHCU equations into word equations. We need the regular constraints to make this encoding sound and ensure that the solutions of the word equations really encode solutions of the corresponding LHCU problem.

In (Levy and Villaret, 2002) it is proved that context unification is reducible to context unification with constants and only one binary symbol. A similar reduction could be applied 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 $X(@(a,b)) \overset{?}{\approx} @(a, X(b))$ into a WU equation could be done using a postorder traversal of the terms of the equation as follows: [3]

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

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:

$$\begin{aligned} \psi_1 &= [W_X \mapsto \epsilon], \\ \psi_2 &= [W_X \mapsto \alpha_@], \end{aligned}$$

where ϵ is the empty word. Notice that $\psi_2(W_X)$ does not correspond to a postorder traversal of a context, while $\psi_1(W_X)$ is the postorder traversal of the empty context $\bullet$. The fact that some of the word unifiers do not correspond to a context unifier leads us to impose regular constraints to this encoding. In what follows we will show that with regular constraints we can get a sound encoding.

We will focus on ground unifiers. Moreover, we will only consider size-minimal ground unifiers, [4] defined as follows.

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

[4] In the study of Context Unification, Word Unification and related problems, it is usual to restrict the study to size-minimal ground unifiers, that in the case of word unification enjoy nice properties like the bound for the exponent of periodicity (see, e.g., Makanin (1977); Kościelski and Pacholski (1996); Schmidt-Schauß and Schulz (1998)). Here, we define minimal unifiers in a different way, without proving that they correspond to the usual definition. This is not important since we only need to ensure that solvable problems have a minimal unifier.

Definition 18. 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 $\{x_1, \ldots, x_n, X_1, \ldots, X_m\} = \mathit{vars}(\rho(\Delta))$ and a is a constant of Σ_{R}.

Notice that, since Σ_{R} contains at least one constant and LHCU is infinitary, any solvable LHCU problem has a minimal unifier. This means that we can reduce the decidability of LHCU to the decidability of the existence of a minimal unifier.

4.1. Properties of LHCU Minimal Unifiers

In this subsection we prove an important property of LHCU minimal unifiers, Lemma 23, that will be used in Subsection 4.3 to reduce LHCU to Word Unification with Regular Constraints. We start with an adaptation of the sound and complete set of rules for Linear Second-Order Unification (Levy, 1996, Definition 4) to the more specialized case of LHCU. We will use this procedure to predict what "shape" minimal unifiers have (see Lemma 21). From the form of these unifiers, we will prove Lemma 23, that is the main ingredient needed to prove decidability of LHCU. Notice that Lemma 23 corresponds to the conjecture described in Levy and Villaret (2001) for general context unification (that has never been proved), but restricted to LHCU. For those interested in general context unification, we describe this correspondence in Subsection 4.2.

Definition 19. The unification procedure for LHCU is described by a set of problem transformations, where every transformation has the form

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

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

Simplification: $s \overset{?}{\approx} s \Longrightarrow \emptyset$,

$@(s_1, s_2) \overset{?}{\approx} @(t_1, t_2) \Longrightarrow \{s_1 \overset{?}{\approx} t_1, s_2 \overset{?}{\approx} t_2\}$,

$X(s) \overset{?}{\approx} X(t) \Longrightarrow \{s \overset{?}{\approx} t\}$,

where $\rho = [\,]$ in all three cases.

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

Imitation: $X(s) \overset{?}{\approx} @(t_1, t_2) \Longrightarrow \{X'(s) \overset{?}{\approx} t_1\}$ and $\rho = [X \mapsto @(X'(\bullet), t_2)]$,

provided that X does not occur in t_2,[5] and X' is fresh.

$x \overset{?}{\approx} s \Longrightarrow \emptyset$ and $\rho = [x \mapsto s]$,

provided that x does not occur in s.[5]

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

where $X \neq Y$ and X' is a fresh context variable.

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

The transformations are applied nondeterministically, starting with $\langle \Delta, [\,] \rangle$, until we get a pair of the form $\langle \emptyset, \varphi \rangle$. Then, $\varphi|_{vars(\Delta)}$, is output as a unifier of Δ.

Proposition 20. *The unification procedure described in Definition 19 is sound and complete:*

Soundness: *If* $\langle \Delta, [\,] \rangle \Longrightarrow^* \langle \emptyset, \varphi \rangle$*, then* φ *is a left-hole context unifier of* Δ*.*
Completeness: *If* φ *is a most general left-hole context unifier of* Δ*, then* $\langle \Delta, [\,] \rangle \Longrightarrow^* \langle \emptyset, \varphi' \rangle$*, for some* φ' *satisfying* $\varphi = \varphi'|_{vars(\Delta)}$*.*[6]

Proof. The proposition is a specialization of Theorems 5 and 6 in (Levy, 1996). Here we only sketch the proof.

The soundness proof is done by induction on the length of the derivation. As induction step, we prove that if $\langle \Delta \cup \{s \overset{?}{\approx} t\}, \varphi \rangle \Longrightarrow \langle \rho(\Delta \cup \Delta'), \rho \circ \varphi \rangle$ and φ' is a unifier of $\rho(\Delta \cup \Delta')$, then $\varphi' \circ \rho$ is a unifier of $\Delta \cup \{s \overset{?}{\approx} t\}$. It is trivial for the simplification rule that transforms a set of equations into another one with exactly the same set of unifiers. The other rules can be decomposed as follows: In the first step an equation E is transformed into $\rho(E)$ (hence, if φ' unifies $\rho(E)$, then $\varphi' \circ \rho$ unifies E) and then one or more simplification steps are made.

The completeness proof has two parts. First, we have to prove that, if φ is a most general unifier of Δ, then there exists a transformation $\langle \Delta, [\,] \rangle \Longrightarrow \langle \Delta', \rho \rangle$, where $\rho \preceq^{vars(\Delta)} \varphi$. This is done by inspection of the rules. This property ensures that there exists a most general unifier φ' of Δ' such that $\varphi(X) = \varphi' \circ \rho(X)$, for any $X \in vars(\Delta)$. We fix a most general unifier φ of Δ, this allows us to construct inductively a sequence of transformations $\langle \Delta, [\,] \rangle \Longrightarrow \langle \Delta_1, \varphi_1 \rangle \Longrightarrow \langle \Delta_2, \varphi_2 \rangle \Longrightarrow \cdots$ where $\varphi_i \preceq^{vars(\Delta)} \varphi$.

Second, we will prove that (for a fixed most general unifier φ) this sequence terminates. The final state is $\langle \emptyset, \varphi' \rangle$, where φ' is a unifier of Δ and $\varphi' \preceq^{vars(\Delta)} \varphi$. Therefore, $\varphi'|_{vars(\Delta)}$ is equal to φ modulo renaming of introduced fresh variables.

The relationship $\Longrightarrow$ is in general not terminating. Therefore, to prove termination of these kinds of transformation sequences requires further arguments. In particular, we will prove that if we require second components, the substitutions φ_i, to satisfy $\varphi_i \preceq^{vars(\Delta)} \varphi$, for a fixed φ, then the sequence cannot be infinite. Intuitively, starting with the identity substitution, the second component *grows* until becoming φ. Therefore, it seems that the size of the partially constructed substitution $\sum_{X \in Dom(\varphi)} |\varphi_i(X)|$ increases with instantiations. However, this is not always true for the projection, that can make a term to decrease. Therefore, we have to measure the size of terms not *counting* the variables that are projected later. We define inductively the size of a term t, or of a

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

substitution φ, w.r.t. a substitution ρ as follows:

$$
\begin{aligned}
&|a|_\rho = |x|_\rho = |\bullet|_\rho = 1\\
&|@(t_1,t_2)|_\rho = |t_1|_\rho + |t_2|_\rho + 1\\
&|X(t)|_\rho = \begin{cases} |t|_\rho & \text{if } \rho(X) = \bullet\\ |t|_\rho + 1 & \text{otherwise}\end{cases}\\
&|\varphi|_\rho = \textstyle\sum_{X\in Dom(\varphi)} |\varphi(X)|_\rho
\end{aligned}
$$

Notice that

$$
\begin{aligned}
|t|_{\tau\circ\rho} &\le |\rho(t)|_\tau\\
|\sigma|_{\tau\circ\rho} &\le |\rho\circ\sigma|_\tau
\end{aligned}
$$

We will compare the states with respect to an order with three components. The first one is the number of remaining variables. Notice that the projection strictly decreases the number of variables, by projecting one of them, whereas imitation and flex-flex introduce one fresh variable and eliminate another one. The second component is the difference between the size of the substitution φ that we want to construct, and the size of the current partially constructed substitution φ_i. Imitation and flex-flex increase the size of the partially constructed substitution, i.e. decrease this component. Notice that the projection rule does not decrease this component when the projected variable does not belong to the domain of φ. This is the reason why the first component is necessary. Finally, since simplification does not modify the partially constructed substitution, we also consider the size of the current problem as a third component.

Formally, we define the size of a state $\langle\Delta_i,\varphi_i\rangle$, with respect to a fixed φ, as the triplet

$$
|\langle\Delta_i,\varphi_i\rangle| = \Big\langle |vars(\Delta_i)|\ ,\ |\varphi| - \sum_{x\in Dom(\varphi)} \big|\varphi_i(x)\big|_\rho\ ,\ \sum_{t\overset{?}{=}u\in\Delta_i} |t| + |u| \Big\rangle
$$

where ρ satisfies $\varphi(X) = \rho\circ\varphi_i(X)$, for all $X \in vars(\Delta)$.[7] Then, the sizes of the states of $\langle\Delta,[\,]\rangle \Longrightarrow \langle\Delta_1,\varphi_1\rangle \Longrightarrow \langle\Delta_2,\varphi_2\rangle \Longrightarrow \cdots$, compared using a lexicographic ordering, strictly decrease. We can easily see that the simplification decreases the sizes of the equations (the third component) not increasing the previous components, the projection the number of variables (first component), and the imitation and flex-flex, increase $\big|(\varphi_i|_{vars(\Delta)})\big|_\rho$, not increasing the number of variables.

The most intuitive of the three components is the second one. It measures how far is the partially computed unifier φ_i from the final most general unifier φ. $\square$

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

The following Lemma describes the "shape" of minimal LHCU unifiers.

Lemma 21. *Given a LHCU equation $s \overset{?}{\approx} t$, for any minimal unifier φ, and any context variable $X \in vars(s \overset{?}{\approx} t)$, we have $\varphi(X) = @(\dots @(\bullet,\varphi(t_n))\dots,\varphi(t_1))$, and for any first-order variable $x \in vars(s \overset{?}{\approx} t)$, we have $\varphi(x) = @(\dots @(a,\varphi(t_n))\dots,\varphi(t_1))$, where*

[7] Notice that the size of a state does not depend on the selection of the ρ satisfying this property.

a is a first-order constant and 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. We say that a term t *right-occurs:* in a set of equations Δ, if they contain a subterm of the form $@(\_, t)$; in a substitution φ, if there exists a variable $X \in Dom(\varphi)$ such that $\varphi(X)$ contains a subterm of the form $@(\_, t)$; and in a state $\langle \Delta, \varphi \rangle$, if it right-occurs in Δ or in φ.

Now, we will prove that right-occurrences are preserved by the transformation rules: If $\langle \Delta_1, \varphi \rangle \Longrightarrow \langle \Delta_2, \rho \circ \varphi \rangle$ and t right-occurs in $\langle \Delta_2, \rho \circ \varphi \rangle$, then there exists a term t' right-occurring in $\langle \Delta_1, \varphi \rangle$ such that $t = \rho(t')$.

For the simplification rule, the preservation is trivial. For the other rules, we can decompose them as an instantiation (I) followed by one or more simplifications (S):

$$\langle \Delta \cup \{s \overset{?}{\approx} t\}, \varphi \rangle \Longrightarrow_{\mathsf{I}} \langle \rho(\Delta \cup \{s \overset{?}{\approx} t\}), \rho \circ \varphi \rangle \Longrightarrow_{\mathsf{S}}^{+} \langle \rho(\Delta \cup \Delta'), \rho \circ \varphi \rangle$$

For the instantiations, we can prove that:

(1) If t right-occurs in $\rho(\Delta)$, then either t right-occurs in ρ, or there exists a t' right-occurring in Δ such that $t = \rho(t')$.

(2) If t right-occurs in $\rho \circ \varphi$, then either t right-occurs in ρ, or there exists a t' right-occurring in φ such that $t = \rho(t')$.

Using these two statements, and since ρ does not contain right-occurring terms in the cases of the projection, the flex-flex, and the second imitation rules, the result holds for these rules.

Finally, for the first imitation rule, $\rho = [X \mapsto @(X'(\bullet), t_2)]$ contains the right-occurrence of t_2, but this is a term right-occurring in the equation $X(s) \overset{?}{=} @(t_1, t_2)$.

By induction on the length of the transformation we can prove the same result for any transformation sequence, in particular, for $\langle \Delta, [\,] \rangle \Longrightarrow^{*} \langle \emptyset, \varphi \rangle$. Therefore, using Proposition 20, for any most general unifier φ of Δ, if t right-occurs in φ, then there exists a right-occurrence of some term t' in Δ such that $t = \varphi(t')$.

By definition of minimal unifier, we can prove that instances of variables have the form $\varphi(X) = @(\dots @(\bullet, \varphi(t_n)) \dots, \varphi(t_1))$ or $\varphi(x) = @(\dots @(a, \varphi(t_n)) \dots, \varphi(t_1))$. Now, since a minimal unifier is the result of the composition of a most general unifier and another substitution that does not introduce new right-occurrences, and the t_i's are right-occurrences in φ, we can conclude the statement of the lemma. $\square$

The previous lemma allows us to prove that, if φ is a minimal unifier of $s \overset{?}{\approx} t$, 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 $s \overset{?}{\approx} t$.

Definition 22. The *number of right accumulated branches (rab)* of a ground term $t \in \mathcal{T}(\{@\} \cup \Sigma_0, \emptyset)$, denoted $\mathrm{rab}(t)$, is defined as:

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

for any $a \in \Sigma_0$ and $t_1, t_2 \in \mathcal{T}(\{@\} \cup \Sigma_0, \emptyset)$.

Lemma 23. *Let $s \overset{?}{\approx} t$ be a LHCU equation and φ a minimal unifier, then*

$$\mathrm{rab}(\varphi(s)) \leq |s| + |t|.$$

Proof. Lemma 21 says that, for any first-order variable x [context variable X], terms occurring in $\varphi(x)$ [in $\varphi(X)$, resp.] as second arguments of an @ have the form $\varphi(t')$ where t' is a subterm of $s \overset{?}{\approx} t$ occurring as a second argument of an @. Therefore, all subterms of $\varphi(s)$ occurring as a second argument of an @ also satisfy this property. Since there are $|s| + |t|$ subterms t' in $s \overset{?}{\approx} t$, and we cannot repeat the same subterm in a branch, $\mathrm{rab}(\varphi(s)) \leq |s| + |t|$. $\square$

4.2. Relationship Between Rab and Strahler Number in CU

As we mentioned, Lemma 23 is a specialization for LHCU of a conjecture stated in Levy and Villaret (2001), that in case of being true, would imply decidability of Context Unification. Here we look into details of this relationship. Readers not interested in it can skip this subsection since it is not necessary in order to prove decidability of LHCU.

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

Definition 24. The *Strahler Number* of a term t built up from binary and nullary symbols, noted $\mathrm{Strahler}(t)$, is defined recursively as follows:

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

for any constant a and binary symbol f.

It is easy to prove that $\mathrm{Strahler}(t) \leq \mathrm{rab}(t)$, for any term t. This implies the following result.

Lemma 25. *For each minimal unifier φ of an LHCU equation $s \overset{?}{\approx} t$ the inequality $\mathrm{Strahler}(\varphi(t)) \leq |s| + |t|$ holds.*

In Levy and Villaret (2001), it is proved that Context Unification is decidable if, and only if, there exists a computable upper bound for the Strahler number of some unifier of every solvable CU problem. Here, Lemma 25 proves that this upper bound exists in the particular case of LHCU. Therefore, we can conclude decidability of LHCU from a small modification of the results of Levy and Villaret (2001). 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 in next subsection is then an alternative proof for the decidability of LHCU based on some ideas of Levy and Villaret (2001).

Now we will define the *flattening* function that will allow us to transform context equations into word equations. Then, we will prove that traversals of rab-bound ground terms are regular. Finally, we will show how to reduce LHCU to Word Unification with regular constraints.

Definition 26. Given a ranked signature $\Sigma_{\mathsf{R}} = \Sigma_0 \cup \Sigma_2$, where $\Sigma_0 \neq \emptyset$ and $\Sigma_2 = \{@\}$, the *flattened alphabet* $\Sigma_{\mathsf{R}}^{\mathcal{F}}$ contains a letter α_f, for each symbol $f \in \Sigma_{\mathsf{R}}$. Similarly, given a set of variables $\mathcal{X}$, $\mathcal{X}^{\mathcal{F}}$ has a distinct word variable W_z for each first-order or context variable $z \in \mathcal{X}$.

The *flattening function* $\mathcal{F}$ over ranked terms, left-hole contexts and context equations is defined as follows:

$$\begin{aligned}
\mathcal{F}(a) &= \alpha_a, \\
\mathcal{F}(@(t_1, t_2)) &= \mathcal{F}(t_1)\,\mathcal{F}(t_2)\,\alpha_@, \\
\mathcal{F}(x) &= W_x, \\
\mathcal{F}(X(t)) &= \mathcal{F}(t)\,W_X, \\
\mathcal{F}(\bullet) &= \epsilon, \\
\mathcal{F}(s \overset{?}{\approx} t) &= \mathcal{F}(s) \overset{?}{=}_w \mathcal{F}(t),
\end{aligned}$$

where a is a constant and $@$ is the binary function symbol of Σ_{R} and α_a and $\alpha_@$ the corresponding letters in $\Sigma_{\mathsf{R}}^{\mathcal{F}}$, x is a first-order and X a context variable in $\mathcal{X}$ and W_x and W_X the corresponding word variables in $\mathcal{X}^{\mathcal{F}}$, and ϵ is the empty word.

We extend the definition of the flattening function $\mathcal{F}$ to transform left-hole FC-substitutions into word substitutions as follows: for an FC-substitution ρ the corresponding $\mathcal{F}(\rho)$ is defined as the substitution that, for each variable $z \in \mathit{Dom}(\rho)$, maps $\mathcal{F}(z)$ into $\mathcal{F}(\rho(z))$.

Example 27. Using this definition we get for instance that the flattening of $\rho = [x \mapsto @(a, b), X \mapsto @(\bullet, b)]$ is $\mathcal{F}(\rho) = [W_x \mapsto \alpha_a\,\alpha_b\,\alpha_@, W_X \mapsto \ \alpha_b\,\alpha_@]$.

Remark 28. Notice that the words resulting from the flattening of terms encode the postorder traversals of these terms. For any ranked term t and context C, we have $\mathcal{F}(C[t]) = \omega_1\,\mathcal{F}(t)\,\omega_2$ for some words ω_1 and ω_2. However, when C is a left-hole context, we have $\mathcal{F}(C[t]) = \mathcal{F}(t)\,\omega$. With the translation $\mathcal{F}(X(t)) = \mathcal{F}(t)\,W_X$, and the induction step $\mathcal{F}\big(\rho(t)\big) = \mathcal{F}(\rho)\big(\mathcal{F}(t)\big)$, we ensure that

$$\begin{aligned}
\mathcal{F}\Big(\rho\big(X(t)\big)\Big) &= \mathcal{F}\Big(\rho(X)\big[\rho(t)\big]\Big) = \mathcal{F}\big(\rho(t)\big)\,\mathcal{F}\big(\rho(X)\big) \\
&= \mathcal{F}(\rho)\Big(\mathcal{F}(t)\,\mathcal{F}(X)\Big) = \mathcal{F}(\rho)\Big(\mathcal{F}\big(X(t)\big)\Big).
\end{aligned}$$

If contexts were not left-hole, then, to obtain the same commutativity result, we would have to define $\mathcal{F}(X(t)) = W'_X\,\mathcal{F}(t)\,W''_X$, using *two* word variables W'_X and W''_X for each context variable X.

Lemma 29. *For any FC-substitution ρ and for any ranked term t (or context) over Σ_{R} and $\mathcal{X}$, we have that $\mathcal{F}(\rho)\big(\mathcal{F}(t)\big) = \mathcal{F}\big(\rho(t)\big)$.*

Proof. By structural induction on t. See previous remark. $\square$

Lemma 30. *If ρ is a unifier of the context unification problem Δ over Σ_{R} and $\mathcal{X}$, then $\mathcal{F}(\rho)$ is a unifier of the Word Unification problem $\mathcal{F}(\Delta)$ over $\Sigma_{\mathsf{R}}^{\mathcal{F}}$ and $\mathcal{X}^{\mathcal{F}}$.*

Proof. It is an easy consequence of Lemma 29. $\square$

Lemma 30 ensures that, if the original LHCU problem is solvable, then the word unification problem resulting from its flattening translation is also solvable. However, similarly to the curryfication described in Section 3, the converse is not true: some solutions of the word equation may not correspond to translations of context unifiers. To avoid this problem, we add regular constraints to the word equations. The reduction of LHCU is then to *Word Unification with Regular Constraints.* These problems consist of a set of word equations of the form $\omega_1 \stackrel{?}{=} \omega_2$, where $\omega_1, \omega_2 \in (\Sigma \cup \mathcal{W})^*$, and a set of regular constraints of the form $\omega \in R$, where $\omega \in (\Sigma \cup \mathcal{W})^*$ and $R \subseteq \Sigma^*$ is a regular language, and Σ is a finite set of letters and $\mathcal{W}$ a countable set of word variables. A solution is a substitution $\psi : \mathcal{W} \to \Sigma^*$ satisfying $\psi(\omega_1) = \psi(\omega_2)$ for any equation $\omega_1 \stackrel{?}{=} \omega_2$, and $\psi(\omega) \in R$ for any regular constraint $\omega \in R$. The problem of deciding whether a word unification problem with regular constraints has a solution was proved to be decidable in (Schulz, 1990). The regular constraints will be used to ensure that the instances of word variables are in the image of the function $\mathcal{F}$, i.e. postorder traversals of terms. Given a ranked signature $\Sigma_{\mathsf{R}} = \Sigma_0 \cup \Sigma_2$, where $\Sigma_0 \neq \emptyset$ and $\Sigma_2 = \{@\}$, the set of traversals of terms define a context-free language, described by the grammar $S \to \alpha_{f_1} \mid \ldots \mid \alpha_{f_r} \mid S\, S\, \alpha_{@}$. However, this language is not regular. Fortunately, the set of traversals of rab-bounded terms *is* a regular language (see Lemma 32), and according to Lemma 23, we only need to consider these classes of terms if we restrict the search for minimal unifiers.

Definition 31. Given a ranked signature $\Sigma_{\mathsf{R}} = \Sigma_0 \cup \Sigma_2$, where $\Sigma_0 \neq \emptyset$ and $\Sigma_2 = \{@\}$, we define the following family of regular languages on the alphabet $\Sigma_{\mathsf{R}}^{\mathcal{F}}$:

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

Lemma 32. *The language L^n defines the set of postorder traversals of ground terms $t \in \mathcal{T}(\Sigma_{\mathsf{R}}, \emptyset)$ satisfying* $\mathrm{rab}(t) \leq n$. *In other words,*
(a) *for any $t \in \mathcal{T}(\Sigma_{\mathsf{R}}, \emptyset)$, if* $\mathrm{rab}(t) \leq n$, *then $\mathcal{F}(t) \in L^n$,*
(b) *for any $\omega \in L^n$, there exists $t \in \mathcal{T}(\Sigma_{\mathsf{R}}, \emptyset)$, such that $\mathcal{F}(t) = \omega$ and* $\mathrm{rab}(t) \leq n$.

Proof.
(a) We proceed by induction on n and structural induction on t. For $n = 0$ it is trivial. For $n > 0$, let $t = @(t_1, t_2)$ and $\mathrm{rab}(t) \leq n+1$. Notice that $\mathrm{rab}(t_1) \leq n+1$ and $\mathrm{rab}(t_2) \leq n$, hence by induction hypothesis $\mathcal{F}(t_1) \in L^{n+1}$ and $\mathcal{F}(t_2) \in L^n$. Therefore, $\mathcal{F}(t) = \mathcal{F}(t_1)\mathcal{F}(t_2)\alpha_{@} \in L^{n+1}L^n\alpha_{@} \subseteq L^{n+1}$.

(b) The existence of t is proved by induction on n. The base case is trivial. For the inductive case, any $\omega \in L^{n+1} = L^0\,(L^n\,\alpha_@)$ may be decomposed as $\omega = \alpha_f\,\omega_1\,\alpha_@ \dots \omega_k\,\alpha_@$, where $\omega_i \in L^n$. By induction hypothesis, we can find $t_i \in \mathcal{T}(\Sigma_\mathsf{R}, \emptyset)$ with $\omega_i = \mathcal{F}(t_i)$. We can construct $t = @(\dots @(f, t_1)\dots, t_k)$ satisfying $\mathcal{F}(t) = \omega$.

We also prove $\mathrm{rab}(t) \le n$ by induction on n and structural induction on t. If $t = @(t_1, t_2)$ and $\mathcal{F}(t) \in L^{n+1}$, then, by definition of L^{n+1}, we have $\mathcal{F}(t_1) \in L^{n+1}$ and $\mathcal{F}(t_2) \in L^n$. By induction $\mathrm{rab}(t_1) \le n+1$ and $\mathrm{rab}(t_2) \le n$. Hence, $\mathrm{rab}(t) \le n+1$. $\square$

Theorem 33. *LHCU is P-reducible to WU with regular constraints.*

Proof. We define the translation of LHCU problems into word unification problems with regular constraints as follows.

Given a LHCU problem Δ, we define its translation $\mathcal{F}^{\mathrm{ext}}(\Delta)$ as the set consisting of the set of word equations $\mathcal{F}(\Delta)$, the regular constraints $W_x \in L^{|\Delta|}$ for every first-order variable x of Δ, and the regular constraint $\alpha_a\,W_X \in L^{|\Delta|}$ for every context variable X of Δ, where a is any of the constants of Σ_R. This translation is polynomial.

We have to prove that Δ is solvable iff $\mathcal{F}^{\mathrm{ext}}(\Delta)$ is solvable.

($\Rightarrow$) Lemma 30 ensures that, if ρ is a solution of Δ, then $\mathcal{F}(\rho)$ is a solution of the word equations $\mathcal{F}(\Delta)$. Assume without loss of generality that ρ is a minimal unifier. For any first-order variable x occurring in Δ, by Lemma 23, we have $\mathrm{rab}(\rho(x)) \le |\Delta|$, since x occurs in some equation $s \overset{?}{\approx} t \in \Delta$, $\rho(x)$ is a subterm of $\rho(s)$ with smaller rab number, and $|s| + |t| \le |\Delta|$. Similarly, $\mathrm{rab}(\rho(X(a))) \le |\Delta|$ for any context variable X occurring in Δ. Now, Lemma 32 entails $\mathcal{F}(\rho(x)) \in L^{|\Delta|}$ and $\mathcal{F}(\rho(X(a))) \in L^{|\Delta|}$, hence $\mathcal{F}(\rho)$ is also a solution of the regular constraints $\mathcal{F}(x) = W_x \in L^{|\Delta|}$ and $\mathcal{F}(X(a)) = a\,W_X \in L^{|\Delta|}$.

($\Leftarrow$) For any solution ψ of $\mathcal{F}^{\mathrm{ext}}(\Delta)$, since it satisfies $\psi(W_x) \in L^{|\Delta|}$ and $\psi(a\,W_X) \in L^{|\Delta|}$, by Lemma 32, there exist a term t_x such that $\mathcal{F}(t_x) = \psi(W_x)$ and a left-hole context C_X such that $\mathcal{F}(C_X[a]) = \psi(a\,W_X)$. We define the substitution ρ mapping x to t_x and X to C_X. Trivially, $\mathcal{F}(\rho) = \psi$. Therefore, by Lemma 29, $\psi(\mathcal{F}(t)) = \mathcal{F}(\rho)(\mathcal{F}(t)) = \mathcal{F}(\rho(t))$, for any term t. In particular, for any equation $s \overset{?}{\approx} t \in \Delta$, since $\psi(\mathcal{F}(s)) = \psi(\mathcal{F}(t))$, we have $\mathcal{F}(\rho(s)) = \mathcal{F}(\rho(t))$. Finally, since $\mathcal{F}$ is injective, we have $\rho(s) = \rho(t)$ and, hence, ρ solves Δ. $\square$

Corollary 34. *Left-Hole Context Unification is decidable.*

Corollary 35. *Sequence Unification is decidable.*

With the following example we illustrate the whole reduction process from sequence equations to word equations with regular constraints and from the resulting word unifiers to sequence unifiers.

Example 36. Consider the sequence unification problem $\{f(U, V) \overset{?}{=} f(a, f(U))\}$. To reduce it to word unification with regular constraints we have to get its curry form and then to flatten the resulting context equation obtaining the corresponding word equation with regular constraints:

$$\Big\{f(U,V) \overset{?}{=} f(a,f(U))\Big\}$$

$$\Downarrow \mathcal{C}$$

$$\Big\{X_V(X_U(a_f)) \overset{?}{\approx} @(@(a_f,a_a),X_U(a_f))\Big\}$$

$$\Downarrow \mathcal{F}^{\mathrm{ext}}$$

$$\left\{\begin{array}{rcl} \alpha_{a_f}\, W_{X_U}\, W_{X_V} & \overset{?}{=}_w & \alpha_{a_f}\,\alpha_{a_a}\,\alpha_{a_@}\,\alpha_{a_f}\, W_{X_U}\,\alpha_{a_@} \\ \alpha_{a_f}\, W_{X_V} & \in & L^9 \\ \alpha_{a_f}\, W_{X_U} & \in & L^9 \end{array}\right\}$$

Some unifiers of the word equation and the regular constraints are:

$$\psi_1 = [\ W_{X_U} \mapsto \epsilon, \qquad W_{X_V} \mapsto \alpha_{a_a}\,\alpha_{a_@}\,\alpha_{a_f}\,\alpha_{a_@}\ ],$$
$$\psi_2 = [\ W_{X_U} \mapsto \alpha_{a_a}\,\alpha_{a_@},\ W_{X_V} \mapsto \alpha_{a_f}\,\alpha_{a_a}\,\alpha_{a_@}\,\alpha_{a_@}\ ].$$

They correspond to the traversal of the left-hole context unifiers:

$$\varphi_1 = [\ X_U \mapsto \bullet, \qquad X_V \mapsto @(@(\bullet,a_a),a_f)\ ],$$
$$\varphi_2 = [\ X_U \mapsto @(\bullet,a_a)\ X_V \mapsto @(\bullet,@(a_f,a_a))\ ]$$

that, in turn, correspond to the curryfication of the sequence unifiers:

$$\sigma_1 = [\ U \mapsto \langle\ \rangle,\quad V \mapsto \langle a,f\rangle\quad],$$
$$\sigma_2 = [\ U \mapsto \langle\, a\,\rangle,\ V \mapsto \langle\, f(a)\,\rangle\ ].$$

Notice that the substitution $\psi_3 = [W_{X_U} \mapsto \alpha_{a_a}\,\alpha_{a_@}\,\alpha_{a_f}, W_{X_V} \mapsto \alpha_{a_a}\,\alpha_{a_@}\,\alpha_{a_f}\,\alpha_{a_@}]$ solves the word equation $\alpha_{a_f}\, W_{X_U}\, W_{X_V} \overset{?}{=}_w \alpha_{a_f}\,\alpha_{a_a}\,\alpha_{a_@}\,\alpha_{a_f}\, W_{X_U}\,\alpha_{a_@}$ but does not satisfy the regular constraint $\alpha_{a_f}\, W_{X_U} \in L^9$ because $\alpha_{a_f}\,\alpha_{a_a}\,\alpha_{a_@}\,\alpha_{a_f} \notin L^9$.

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 stronger: decidability of unification for an extension of SU: *Extended Sequence Unification* (ESU). This extension 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, for instance, $@(x_v,a)$ and $X_V(x_v)$ that could be considered as the curryfication of $v(a)$ and $v(V)$.

To define ESU more formally, we extend the notion of unranked terms over Σ_{U} and $\mathcal{V} = \mathcal{V}_{\mathsf{I}} \cup \mathcal{V}_{\mathsf{S}}$:

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

where $v \in \mathcal{V}_{\mathsf{I}}$, $V \in \mathcal{V}_{\mathsf{S}}$, $f \in \Sigma_{\mathsf{U}}$, and $n \geq 0$.

We denote the set of such extended terms with $\mathcal{T}^{\mathsf{e}}_{\mathsf{U}}(\Sigma_{\mathsf{U}},\mathcal{V})$ and use l and r to denote its elements. Terms of the form $v()$ and $f()$ are abbreviated respectively by v and f. Extended individual terms are terms from $\mathcal{T}^{\mathsf{e}}_{\mathsf{U}}(\Sigma_{\mathsf{U}},\mathcal{V}) \setminus \mathcal{V}_{\mathsf{S}}$.

We define the following operation on terms and sequences of terms.

Definition 37. Given an individual term $r \in \mathcal{T}^{\mathsf{e}}_{\mathsf{U}}(\Sigma_{\mathsf{U}},\mathcal{V}) \setminus \mathcal{V}_{\mathsf{S}}$ and a sequence of terms $\langle r_1,\dots,r_n\rangle \in \mathcal{T}^{\mathsf{e}}_{\mathsf{U}}(\Sigma_{\mathsf{U}},\mathcal{V})$ the operation app is defined by

$$\begin{aligned}
\mathrm{app}(f(r'_1,\dots,r'_m),\langle r_1,\dots,r_n\rangle) &= f(r'_1,\dots,r'_m,r_1,\dots,r_n),\\
\mathrm{app}(v(r'_1,\dots,r'_m),\langle r_1,\dots,r_n\rangle) &= v(r'_1,\dots,r'_m,r_1,\dots,r_n).
\end{aligned}$$

Notice that this operation is defined for *any* individual term and *any* sequence. In particular, $\mathrm{app}(f,f) = \mathrm{app}(f(),\langle f\rangle) = f(f)$.

Application of IS-substitution to an extended unranked term has to take into account the term application operation. We extend the definition of term instantiation as follows:

$$\sigma(v(r_1,\dots,r_n)) = \mathrm{app}(\sigma(v),\langle\sigma(r_1),\dots,\sigma(r_n)\rangle).$$

For instance, if $\sigma = [v \mapsto f(a,b)]$, then $\sigma(v(c,v)) = f(a,b,c,f(a,b))$.

Now, ESU problems can be defined as a set of equations (unoriented pairs) of individual extended terms, denoted $\{l_1 \overset{?}{=} r_1,\dots,l_n \overset{?}{=} r_n\}$. The notions of unifier, solvability, etc. carry over from SU.

Currying transformation can be extended in a straightforward way to $\mathcal{T}^{\mathsf{e}}_{\mathsf{U}}(\Sigma_{\mathsf{U}},\mathcal{V})$:

Definition 38. Definition 4 is extended to $\mathcal{T}^{\mathsf{e}}_{\mathsf{U}}(\Sigma_{\mathsf{U}},\mathcal{V}) \to \mathcal{T}(\Sigma^{\mathcal{C}}_{\mathsf{U}},\mathcal{V}^{\mathcal{C}})$ by adding the rules:

$$\begin{aligned}
\mathcal{C}(v(r_1,\dots,r_n,V)) &= X_V(\mathcal{C}(v(r_1,\dots,r_n)))\\
\mathcal{C}(v(r_1,\dots,r_n)) &= @(\mathcal{C}(v(r_1,\dots,r_{n-1})),\mathcal{C}(r_n)) \qquad \text{where } r_n \notin \mathcal{V}_{\mathsf{S}}.
\end{aligned}$$

With this extension, any ranked term from $\mathcal{T}(\Sigma^{\mathcal{C}}_{\mathsf{U}},\mathcal{V}^{\mathcal{C}})$ becomes legal, and we can prove the following result.

Lemma 39. *The extended function $\mathcal{C} : \mathcal{T}^{\mathsf{e}}_{\mathsf{U}}(\Sigma_{\mathsf{U}},\mathcal{V}) \to \mathcal{T}(\Sigma^{\mathcal{C}}_{\mathsf{U}},\mathcal{V}^{\mathcal{C}})$ is bijective and preserves solvability of problems.*

Proof. The inverse of the function satisfies:

$$\begin{aligned}
\mathcal{C}^{-1}(a_f) &= f,\\
\mathcal{C}^{-1}(x_v) &= v,\\
\mathcal{C}^{-1}(@(s,t)) &= \mathrm{app}(\mathcal{C}^{-1}(s),\mathcal{C}^{-1}(t)),\\
\mathcal{C}^{-1}(X_V(t)) &= \mathrm{app}(\mathcal{C}^{-1}(t),V).
\end{aligned}$$

With this equations, proof of bijectivity is straightforward.

The preservation of solvability can be proved by techniques similar to the ones described in Section 3. $\square$

Since this extension of the curryfication preserves solvability, and it is bijective, $\mathcal{C}^{-1}$ also preserves solvability, and we *close the circle* with the following corollaries.

Corollary 40. *Extended Sequence Unification (ESU) and Left-Hole Context Unification (LHCU) are mutually P-reducible problems.*

Corollary 41. *Extended Sequence Unification is decidable.*

5.1. Complexity Results

If one of the sides of each equation in an ESU problem is ground, then we have an *Extended Sequence Matching* (ESM) problem. This problem is NP-complete. It can be shown—similarly to NP-completeness of Context Matching (CM) (Schmidt-Schauß and Schulz, 1998)—by reduction of *positive 1-IN-3-SAT* (Garey and Johnson, 1979) to ESM. A positive 1-IN-3-SAT problem is given by a set of clauses $\{C_1, \dots, C_m\}$ where each clause C_i contains exactly three positive literals $x_{p(i,1)} \vee x_{p(i,2)} \vee x_{p(i,3)}$ from a set of literals $\{x_1, \dots, x_n\}$. A truth assignment solves the problem if it maps exactly one literal from each clause to true.

We describe two different reductions:

Reduction 1. We represent truth values as a (for `false`) and $f(a)$ (for `true`). We have an individual variable v_i and a sequence variable V_i for each clause C_i, and an individual variable u_j for each literal x_j. We encode each clause $C_i = x_{p(i,1)} \vee x_{p(i,2)} \vee x_{p(i,3)}$ as an ESM equation

$$v_i(g(u_{p(i,1)}, u_{p(i,2)}, u_{p(i,3)}), V_i) \stackrel{?}{=} h(g(f(a), a, a), g(a, f(a), a), g(a, a, f(a))).$$

This encoding proves NP-completeness of solvability of ESM problems where each sequence variable and each head-position individual variable occurs at most once, and the number of occurrences of other individual variables is not restricted. We can refine this encoding as follows.

Reduction 2. For the set of clauses $\{x_{p(i,1)} \vee x_{p(i,2)} \vee x_{p(i,3)}\}_{i=1,\dots,m}$, we have an individual variable v_i and a sequence variable V_i for each clause C_i with $i = 1, \dots, m$, an individual variable u_j and a sequence variable U_j for each literal x_j with $j = 1, \dots, n$, and an individual variable w_i^k for each occurrence of a literal, with $i = 1, \dots, m$ and $k = 1, 2, 3$.

For each clause C_i, with $i = 1, \dots, m$, we have the equation

$$v_i(g(w_i^1, w_i^2, w_i^3), V_i) \stackrel{?}{=} h(g(f(a), a, a), g(a, f(a), a), g(a, a, f(a))).$$

To ensure that all occurrences of the same literal get the same value, we add the following equations. For each literal x_j, with $j = 1, \dots, n$, we have the equation

$$u_j(g(w_{k_1}^{k'_1}, \dots, w_{k_{n_j}}^{k'_{n_j}}), U_j) \stackrel{?}{=} h(g(a, \dots, a), g(f(a), \dots, f(a))),$$

where literal x_j occurs at position k'_r of clause k_r, for the n_j occurrences of x_j, i.e. $p(k_r, k'_r) = j$, for $r = 1, \dots, n_j$.

In this second encoding, the first type of equations set exactly one of $w_i^1 \vee w_i^2 \vee w_i^3$ equal to $f(a)$. The second type of equations set n_j different individual variables to the same truth value, where n_j is the number of times that x_j occurs in the instance of the problem. It is easy to see that the individual variables corresponding to literal occurrences occur twice, once in an equation of the first type and once in an equation of the second type.

Theorem 42. *Extended Sequence Matching where each variable occurs at most twice (*Varity 2 ESM*) is NP-complete.*

Compare this encoding with the proof of NP-completeness of Varity 2 Context Matching (Schmidt-Schauß and Stuber, 2004). In fact, what we show with this encoding is NP-completeness of the fragment of ESM with linear sequence and head-position individual variables and at most two occurrences of the other individual variables.

Currying transformation (bijectively) maps ESM to *Left-Hole Context Matching* (LHCM). This allows us to transfer some of the other complexity results for Context Matching (Schmidt-Schauß and Stuber, 2004) to ESM, because the proofs for the fragments of Context Matching in (Schmidt-Schauß and Stuber, 2004) are also valid for LHCM:

- *Linear ESM* (the fragment where each variable occurs at most once) is in $O(n^3)$ where n is the size of the problem. It follows from the same result for *Linear Context Matching.*
- *Shared Linear Context Matching* (SLCM) is a fragment of Context Matching where all occurrences of the same context variable are applied to the same term. Inverse currying transforms SLCM into the 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 $f_1, f_2 \in \Sigma_{\mathsf{U}}$, 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 are the same. SLCM is in P, therefore PCESM is in P.

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

5.2. Unification Procedure for ESU

In the rest of the section we will be concerned with computing the solution set for ESU problems. They, obviously, can be solved by the procedure in Definition 19, first transforming ESU problems into LHCU problems and then translating the unifiers back. The other way is to extend the solving procedure for SU (Kutsia, 2007) with the rules for head-position individual variables and obtain a procedure for ESU. In this way we get a complete solving procedure that can be directly applied to ESU problems without transforming them to LHCU. Below we will follow this idea and formulate transformation rules for ESU.

To make the paper self-contained, we put here the relevant (slightly modified) SU transformation rules from (Kutsia, 2007), which, for uniformity reasons, are formulated similarly to the rules from Definition 19, and add two new rules for head-position individual variables. The notation $\tilde{r}$ below stands for a sequence of extended unranked terms, and f is either a function symbol or an individual variable (occurring in the functional position).

Definition 43. The unification procedure for ESU is described by the set of problem transformations, where every transformation has the form

$$\langle \Delta \cup \{l \stackrel{?}{=} r\}, \vartheta\rangle \Longrightarrow \langle \sigma(\Delta \cup \Delta'), \sigma \circ \vartheta\rangle$$

and is characterized by a rule $l \stackrel{?}{=} r \Longrightarrow \Delta'$ and a substitution σ.

Simplification: $r \overset{?}{=} r \Longrightarrow \emptyset$,

$\mathsf{f}(l, \tilde{l}) \overset{?}{=} \mathsf{f}(r, \tilde{r}) \Longrightarrow \{l \overset{?}{=} r, \mathsf{f}(\tilde{l}) \overset{?}{=} \mathsf{f}(\tilde{r})\}$, $l, r \notin \mathcal{V}_\mathsf{S}$,

$\mathsf{f}(V, \tilde{l}) \overset{?}{=} \mathsf{f}(V, \tilde{r}) \Longrightarrow \{\mathsf{f}(\tilde{l}) \overset{?}{=} \mathsf{f}(\tilde{r})\}$,

where $\sigma = [\,]$ in all three cases.

Projection: $\mathsf{f}(V, \tilde{l}) \overset{?}{=} \mathsf{f}(\tilde{r}) \Longrightarrow \{\mathsf{f}(\tilde{l}) \overset{?}{=} \mathsf{f}(\tilde{r})\}$ and $\sigma = [V \mapsto \langle\rangle]$.

Imitation: $v \overset{?}{=} r \Longrightarrow \emptyset$ and $\sigma = [v \mapsto r]$,

provided that v does not occur in r.[7]

$\mathsf{f}(V, \tilde{l}) \overset{?}{=} \mathsf{f}(r, \tilde{r}) \Longrightarrow \{\mathsf{f}(V', \tilde{l}) \overset{?}{=} \mathsf{f}(\tilde{r})\}$ and $\sigma = [V \mapsto \langle r, V'\rangle]$,

provided that V does not occur in r,[7] and V' is fresh.

Head Variables: $v(\tilde{l}) \overset{?}{=} \mathsf{f}(\tilde{r}) \Longrightarrow \{\mathsf{f}(V', \tilde{l}) \overset{?}{=} \mathsf{f}(\tilde{r})\}$ and $\sigma = [v \mapsto \mathsf{f}(V')]$,

where $v \neq \mathsf{f}$ and V' is fresh.

To solve an ESU problem Δ, we start with $\langle\Delta, [\,]\rangle$ and apply nondeterministically the transformations above until we get a pair of the form $\langle\emptyset, \vartheta\rangle$. Then ϑ is returned as a solution of Δ. To prune the search tree, unsolvable problems are not transformed.

Proposition 44. *The unification procedure described in Definition 43 is sound and complete:*

Soundness: *If* $\langle\Delta, [\,]\rangle \Longrightarrow^* \langle\emptyset, \vartheta\rangle$, *then* ϑ *is a unifier of* Δ.
Completeness: *If* ϑ *is a unifier of* Δ, *then* $\langle\Delta, [\,]\rangle \Longrightarrow^* \langle\emptyset, \vartheta'\rangle$ *where* $\vartheta' \preceq^{vars(\Delta)} \vartheta$.

Proof. Soundness is proved by induction on the length of transformation sequences. As induction step, we prove that if $\langle\Delta \cup \{l \overset{?}{=} r\}, \vartheta\rangle \Longrightarrow \langle\sigma(\Delta \cup \Delta'), \sigma \circ \vartheta\rangle$ and ϑ' is a unifier of $\sigma(\Delta \cup \Delta')$, then $\vartheta' \circ \sigma$ is a unifier of $\Delta \cup \{l \overset{?}{=} r\}$. For the head variables rule this can be easily shown, and for the other rules it follows from the corresponding result in (Kutsia, 2007).

To prove completeness, first we have to construct a sequence of transformations $\langle\Delta, [\,]\rangle \Longrightarrow \langle\Delta_1, \vartheta_1\rangle \Longrightarrow \langle\Delta_2, \vartheta_2\rangle \Longrightarrow \cdots$ where $\vartheta_i \preceq^{vars(\Delta)} \vartheta$. Such a sequence can be constructed recursively. Assume $\langle\Delta_i, \vartheta_i\rangle$ belongs to the sequence. We have to find $\langle\Delta_{i+1}, \vartheta_{i+1}\rangle$ such that $\langle\Delta_i, \vartheta_i\rangle \Longrightarrow \langle\Delta_{i+1}, \vartheta_{i+1}\rangle$ and $\vartheta_{i+1} \preceq^{vars(\Delta)} \vartheta$. There are several cases depending on the form of equation from Δ_i that we are going to transform. We consider here only the case when it has the form $v(\tilde{l}) \overset{?}{=} \mathsf{f}(\tilde{r})$. First assume that f is not a variable. Then we transform the equation with the head variables rule by the substitution $\sigma_i = [v \mapsto \mathsf{f}(V')]$. If f is a variable itself, we compare the lengths len_1 and len_2 of argument sequences respectively in $\vartheta(v)$ and $\vartheta(\mathsf{f})$. If $len_1 \leq len_2$ then we use the head variables rule with the substitution $\sigma_i = [\mathsf{f} \mapsto v(V')]$, otherwise the same rule is used with the substitution $\sigma_i = [v \mapsto \mathsf{f}(V')]$. In all cases $\vartheta_{i+1} = \sigma_i \circ \vartheta_i \preceq^{vars(\Delta)} \vartheta$.

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

The second step in completeness proof is to show that this sequence terminates. We define inductively the size of a term r, a sequence of terms $\langle r_1, \dots, r_n\rangle$, or of a substitution ϑ, with respect to a substitution σ as follows:

$$
\begin{aligned}
&|v|_\sigma = 2.\\
&|V|_\sigma = \begin{cases} 0 \text{ if } \sigma(V) = \langle\rangle, \\ 1 \text{ otherwise.}\end{cases}\\
&|f(r_1, \dots, r_n)|_\sigma = |r_1|_\sigma + \cdots + |r_n|_\sigma + 2.\\
&|\langle r_1, \dots, r_n\rangle|_\sigma = |r_1|_\sigma + \cdots + |r_n|_\sigma.\\
&|\vartheta|_\sigma = \textstyle\sum_{w \in Dom(\vartheta)} |\vartheta(w)|_\sigma.
\end{aligned}
$$

Given ϑ and Δ, we define the size of a pair $\langle \Delta_i, \vartheta_i\rangle$ as the quadruple

$$|\langle \Delta_i, \vartheta_i\rangle| = \langle |hvars(\Delta_i)|,\ |vars(\Delta_i)|,\ |\vartheta|_{[]} - |(\vartheta_i|_{vars(\Delta)})|_\sigma,\ \bigcup_{l \overset{?}{=} r \in \Delta_i} \{|l|_\sigma, |r|_\sigma\}\rangle$$

where σ satisfies $\vartheta(w) = \sigma \circ \vartheta_i(w)$ for all $w \in vars(\Delta)$, and $hvars(\Delta_i)$ stand for the set of variables that occur in a head position in Δ_i. The sizes of the states of $\langle \Delta, []\rangle \Longrightarrow \langle \Delta_1, \vartheta_1\rangle \Longrightarrow \langle \Delta_2, \vartheta_2\rangle \Longrightarrow \cdots$, compared using a lexicographic ordering, and a multiset ordering for the fourth component, strictly decrease: The head variables rule decreases the first component, the projection rule and the first imitation rule decrease the second component without increasing the first one, the second imitation rule decreases either the third or the fourth component without increasing the others before that, and the simplification rules decrease the fourth component without increasing the first three. □

Example 45. The ESU problem $\{f(a, V) \overset{?}{=} v(a, b)\}$ has two most general unifiers: $\sigma_1 = \{V \mapsto b, v \mapsto f()\}$ and $\sigma_2 = \{V \mapsto \langle U, a, b\rangle, v \mapsto f(a, U)\}$. A derivation that computes the substitution that coincides to σ_2 on the set of variables of the problem proceeds as follows:

$$
\begin{aligned}
&\langle \{f(a, V) \overset{?}{=} v(a, b)\}, []\rangle \Longrightarrow\\
&\langle \{f(a, V) \overset{?}{=} f(U', a, b)\}, [v \mapsto f(U')]\rangle \Longrightarrow\\
&\langle \{f(V) \overset{?}{=} f(U, a, b)\}, [v \mapsto f(a, U), U' \mapsto \langle a, U\rangle]\rangle \Longrightarrow\\
&\langle \{f(V') \overset{?}{=} f(a, b)\}, [v \mapsto f(a, U), U' \mapsto \langle a, U\rangle, V \mapsto \langle U, V'\rangle]\rangle \Longrightarrow\\
&\langle \{f(V'') \overset{?}{=} f(b)\}, [v \mapsto f(a, U), U' \mapsto \langle a, U\rangle, V \mapsto \langle U, a, V''\rangle, V' \mapsto \langle a, V''\rangle]\rangle \Longrightarrow\\
&\langle \{f(V''') \overset{?}{=} f()\}, [v \mapsto f(a, U), U' \mapsto \langle a, U\rangle, V \mapsto \langle U, a, b, V'''\rangle,\\
&\qquad\qquad\qquad\qquad V' \mapsto \langle a, b, V'''\rangle, V'' \mapsto \langle b, V'''\rangle]\rangle \Longrightarrow\\
&\langle \{f() \overset{?}{=} f()\}, [v \mapsto f(a, U), U' \mapsto \langle a, U\rangle, V \mapsto \langle U, a, b\rangle,\\
&\qquad\qquad\qquad\qquad V' \mapsto \langle a, b\rangle, V'' \mapsto \langle b\rangle, V''' \mapsto \langle\rangle]\rangle \Longrightarrow\\
&\langle \emptyset, [v \mapsto f(a, U), U' \mapsto \langle a, U\rangle, V \mapsto \langle U, a, b\rangle, V' \mapsto \langle a, b\rangle, V'' \mapsto \langle b\rangle, V''' \mapsto \langle\rangle]\rangle.
\end{aligned}
$$

6. Conclusion

We have studied the relation between two generalizations of Word Unification: Sequence Unification (SU) and Context Unification (CU). We have introduced 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 have added a restriction on the form of solutions of context unification problems, obtaining the left-hole variant of Context Unification. We have proved 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 have also proved that Left-Hole CU is decidable, reducing it to Word Unification with regular constraints. This result gives a decidability proof for an extension of SU, and, in particular, a new proof of decidability of SU. The reduction is reversible, therefore Left-Hole CU and Extended SU are equivalent problems. Based on the transformation, we have transferred some complexity results from context matching to sequence matching. Finally, we have defined a procedure for solving Extended SU problems and have proved its soundness and completeness.

7. Acknowledgements

This research has been partially supported by the CICyT projects SuRoS (ref. TIN2008-04547/TIN) and Mulog-2 (ref. TIN2007-68005-C04-01) and by the EC Framework 6 Programme for Integrated Infrastructures Initiatives under the project SCIEnce (contract No. 026133).

References

Boley, H., 1999. A Tight, Practical Integration of Relations and Functions. Vol. 1712 of LNAI. Springer.

Buchberger, B., Crăciun, A., Jebelean, T., Kovács, L., Kutsia, T., Nakagawa, K., Piroi, F., Popov, N., Robu, J., Rosenkranz, M., Windsteiger, W., 2006. Theorema: Towards computer-aided mathematical theory exploration. J. Applied Logic 4 (4), 470–504.

Chasseur, E., Deville, Y., 1998. Logic program schemas, constraints and semi-unification. In: Proc. of the 7th International Workshop on Logic Program Synthesis and Transformation (LOPSTR'97). Vol. 1463 of LNCS. Springer, pp. 69–89.

Chiba, Y., Aoto, T., Toyama, Y., 2005. Program transformation by templates based on term rewriting. In: Proc. of the 7th International ACM SIGPLAN Conference on Principles and Practice of Declarative Programming (PPDP'05). ACM Press, pp. 59–69.

Coelho, J., Florido, M., 2004. CLP(Flex): Constraint logic programming applied to XML processing. In: Proc. of the OTM Confederated International Conferences, CoopIS, DOA, and ODBASE 2004. Part II. Vol. 3291 of LNCS. Springer, pp. 1098–1112.

Coelho, J., Florido, M., 2006. VeriFLog: A constraint logic programming approach to verification of website content. In: Proc. of the APWeb 2006 International Workshops: XRA, IWSN, MEGA, and ICSE. Vol. 3842 of LNCS. Springer, pp. 148–156.

Coelho, J., Florido, M., Kutsia, T., 2007. Sequence disunification and its application in collaborative schema construction. In: Proc. of the WISE 2007 International Workshops. Vol. 4832 of LNCS. Springer, pp. 91–102.

Common Logic Working Group, 2007. Common Logic Working Group Documents: Common Logic Standard, `http://common-logic.org/`.

Diekert, V., 2002. Makanin's algorithm. In: Algebraic aspects of combinatorics on words. Cambridge University Press, Ch. 12, pp. 342–390.

Garey, M., Johnson, D., 1979. Computers and Intractability. Freeman, San Francisco, CA.

Genesereth, M. R., Petrie, C., Hinrichs, T., Hondroulis, A., Kassoff, M., Love, N., Mohsin, W., 1998. Knowledge Interchange Format, draft proposed American National Standard (dpANS). Tech. Rep. NCITS.T2/98-004.

Ginsberg, M. L., 1991. The MVL theorem proving system. SIGART Bull. 2 (3), 57–60.

Hamana, M., 1997. Term rewriting with sequences. In: Proc. of the First International Theorema Workshop. Technical report 97–20, RISC, Linz, Austria.

Hayes, P., Menzel, C., 2001. Semantics of Knowledge Interchange Format, In: IJCAI 2001 Workshop on the IEEE Standard Upper Ontology.

Hayes, P. J., Menzel, C., 2005. Simple common logic. In: W3C Workshop on Rule Languages for Interoperability. W3C.

Jaffar, J., 1990. Minimal and complete word unification. J. ACM 37 (1), 47–85.

Koller, A., 1998. Evaluating context unification for semantic underspecification. In: 3rd ESSLLI Student Session (ESSLLI'98), August 17-28. pp. 188–199.

Kościelski, A., Pacholski, L., 1996. Complexity of Makanin's algorithm. J. ACM 43 (4), 670–684.

Kutsia, T., 2002. Unification with sequence variables and flexible arity symbols and its extension with pattern-terms. In: Artificial Intelligence, Automated Reasoning and Symbolic Computation. Proc. of Joint AISC'2002–Calculemus'02 Conference. Vol. 2385 of LNAI. Springer, pp. 290–304.

Kutsia, T., 2007. Solving equations with sequence variables and sequence functions. J. Symbolic Computation 42 (3), 352–388.

Kutsia, T., Levy, J., Villaret, M., 2007. Sequence unification through currying. In: Proc. of the 18th International Conference on Rewriting Techniques and Applications (RTA'07). Vol. 4533 of LNCS. Springer, pp. 288–302.

Kutsia, T., Marin, M., 2005. Matching with regular constraints. In: Proc. of the 12th International Conference on Logic in Programming, Artificial Intelligence and Reasoning (LPAR'05). Vol. 3835 of LNAI. Springer, pp. 215–229.

Levy, J., 1996. Linear second-order unification. In: Proc. of the 7th International Conference on Rewriting Techniques and Applications (RTA'96). Vol. 1103 of LNCS. Springer, pp. 332–346.

Levy, J., Niehren, J., Villaret, M., 2005. Well-nested context unification. In: Proc. of the 20th International Conference on Automated Deduction (CADE-20). Vol. 3632 of LNAI. Springer, pp. 149–163.

Levy, J., Villaret, M., 2001. Context unification and traversal equations. In: Proc. of the 12th International Conference on Rewriting Techniques and Applications (RTA'01). Vol. 2041 of LNCS. pp. 169–184.

Levy, J., Villaret, M., 2002. Currying second-order unification problems. In: Proc. of the 13th International Conference on Rewriting Techniques and Applications (RTA'02). Vol. 2378 of LNCS. Springer, pp. 326–339.

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

Marin, M., Kutsia, T., 2006. Foundations of the rule-based system RhoLog. J. Applied Non-Classical Logics 16 (1–2), 151–168.

Niehren, J., Pinkal, M., Ruhrberg, P., 1997. A uniform approach to underspecification and parallelism. In: Proc. of the 35th Annual Meeting of the ACL and the 8th Conference of the European Chapter of the ACL (ACL'97). pp. 410–417.

Niehren, J., Villaret, M., 2002. Parallelism and tree regular constraints. In: Proc. of the 9th International Conference on Logic for Programming, Artificial Intelligence, and Reasoning (LPAR'02). Vol. 2514 of LNCS. Springer, Tbilisi, Georgia, pp. 311–326.

Niehren, J., Villaret, M., 2005. Describing lambda terms in context unification. In: Proc. of the 5th International Conference on Logical Aspects in Computational Linguistics (LACL'05). Vol. 3492 of LNAI. Springer, pp. 221–237.

Okui, S., Suzuki, T., 2006. Pattern matching of incompletely RE-typed expressions via transformation. IPSJ Transactions on Programming 47 (SIG 6 (PRO29)), 37–49.

Paulson, L., 1990. Isabelle: the next 700 theorem provers. In: Odifreddi, P. (Ed.), Logic and Computer Science. Academic Press, pp. 361–386.

Plandowski, W., 1999. Satisfiability of word equations with constants is in PSPACE. In: Proc. of the 40th Annual Symposium on Foundations of Computer Science (FOCS'99). IEEE Press, New York City, USA, pp. 495–500.

Richardson, J., Fuchs, N. E., 1997. Development of correct transformation schemata for Prolog programs. In: Proc. of the 7th International Workshop on Logic Program Synthesis and Transformation (LOPSTR'97). Vol. 1463 of LNCS. Springer, pp. 263–281.

Schmidt-Schauß, M., 2002. A decision algorithm for stratified context unification. J. Logic and Computation 12, 929–953.

Schmidt-Schauß, M., Schulz, K. U., 1998. On the exponent of periodicity of minimal solutions of context equations. In: Proc. of 9th International Conference on Rewriting Techniques and Applications (RTA'98). Vol. 1379 of LNCS. Springer, pp. 61–75.

Schmidt-Schauß M., Stuber, J., 2004. The complexity of linear and stratified context matching problems. Theory of Computing Systems 37 (6), 717–740.

Schulz, K. U., 1990. Makanin's algorithm for word equations – two improvements and a generalization. In: Proc. of the International Workshop on Word Equations and Related Topics (IWWERT'90). Vol. 572 of LNCS. Springer, Tübingen, Germany, pp. 85–150.

Wolfram, S., 2003. The Mathematica Book, 5th Edition. Wolfram Media.