A Decision Algorithm for Stratified Context Unification

Manfred Schmidt-Schauß

Fachbereich Informatik
Johann Wolfgang Goethe-Universität
Postfach 11 19 32
D-60054 Frankfurt, Germany
E-mail:schauss@ki.informatik.uni-frankfurt.de

Abstract. Context unification is a variant of second-order unification and also a generalization of string unification. Currently it is not known whether context unification is decidable. An expressive fragment of context unification is stratified context unification. Recently, it turned out that stratified context unification and one-step rewrite constraints are equivalent.
This paper contains a description of a decision algorithm SCU for stratified context unification together with a proof of its correctness, which shows decidability of stratified context unification as well as of satisfiability of one-step rewrite constraints.

1 Introduction

Context unification is a variant of second-order unification and also a generalization of string unification. There are unification procedures for the more general problem of higher-order unification (see e.g. [Pie73,Hue75,SG89,Pre95]). It is well known that higher-order unification and second-order unification are undecidable [Gol81,Far91,LV00].

String unification was shown to be decidable by Makanin [Mak77]. Recent upper complexity estimations are that it is in EXPSPACE [Gut98], in NEXPTIME [Pla99a] and even in PSPACE [Pla99b].

Context unification problems are restricted second-order unification problems. Context variables represent terms with exactly one hole in contrast to a term with an arbitrary number of (equally named) holes in the general second-order case. The name *contexts* was coined in [Com93]. Currently, it is not known whether context unification is decidable. It is known that it is $\mathcal{NP}$-hard (cf. [SSS98]), and that satisfiability of formulas in a logical theory of context unification is undecidable [NPR97a,Vor98].
There are some decidable fragments: i. If for every context variable X, all occurrences of X have the same argument [Com98a,Com98b]; ii. If the number of occurrences of every first-order variable and context variable is at most two [Lev96]; iii. if there are at most two context variables [SSS99]. In this paper

we show that stratified context unification is decidable, which provides a rather expressive fragment with a decidable unification problem.

A decidable restriction of second-order unification similar in spirit to context unification is bounded second-order unification [SS99], where second-order variables represent terms with at most n number of holes, where n is a positive integer selected beforehand.

Applications of context unification are for example in computational linguistics [NPR97a], in particular as a uniform framework for semantic underspecification of natural language [NPR97b]. The fragment of stratified context unification is expressive enough for applying it in computational linguistics. It was also used in equational unification as an important step in showing decidability of distributive unification [SS98]. Recently it was noticed that one-step rewrite constraints and stratified context unification can be interreduced [NTT00]. The result in this paper then implies that satisfiability of one-step rewrite constraints is decidable. Hence this decidability result is also a contribution to research on one-step rewriting [Tre96,CSTT99]. To my best knowledge, there is no other decidability proof for both problems.

An important motivation for writing this paper is to describe a correct decision algorithm for stratified context unification and also to provide a rigorous proof. Previous descriptions of algorithms and proofs are [SS94,SS95] and [Lev96], where the algorithms are not correct and/or proofs contain gaps or the termination proof is incomplete. The treatment in [SS98] covers only a very restricted signature.

The algorithm SCU that is described in this paper optimizes simplicity of the description and not efficiency. This improves upon [SS95,SS94] insofar as no parametric terms are necessary and that there is no call of a decision algorithm for string unification [Mak77]. It also improves and generalizes the algorithm in [SS98], where the signature is restricted to containing only one non-constant binary function symbol. Instead of syntactically introducing integer exponents for ground contexts, the algorithm SCU uses an n-fold copy. This again simplifies the description of the algorithm sacrificing efficiency.

SCU makes use of a lemma on the exponent of periodicity of a minimal solution of context unification problems, proved in [SSS98], which is a generalization of a similar result for string unification [Mak77,KP96].

An experimental implementation of stratified context unification (with exponents) was done in [Höh97].
The following result is proved in this paper:

Theorem: Stratified context unification is decidable.

A corollary following from [NTT00] is:

Theorem: Satisfiability of one-step rewrite constraints is decidable.

The structure of the paper is as follows. After section 2 on preliminary definitions, section 3 contains an overview of the algorithm SCU, and also a proposition

on the connection between deterministic and non-deterministic unification algorithms. Sections 3 contains the definition of decomposition steps. Sections 4 and 5 contain the definition of the rules and proofs of their correctness.

2 Preliminaries

Let Σ be a signature of function symbols. Every function symbol comes with an arity, denoted $ar(f)$, which is a nonnegative integer. Function symbols with $ar(f) = 0$ are also called *constant symbols*. We assume that the signature contains at least one constant symbol and at least one non-constant function symbol, in particular we allow also that the signature may be infinite or monadic. Let $\mathcal{V}_1$ be the set of first-order variables $x, y, z, \ldots$, $\mathcal{V}_2$ be the set of context variables $X, Y, Z, \ldots$, and $\mathcal{V} := \mathcal{V}_1 \cup \mathcal{V}_2$.
Terms t are formed using the grammar

$$t ::= x \mid f(t_1, \ldots, t_{ar(f)}) \mid X(t_0),$$

where x is a first-order variable, f is a function symbol, X is a context variable, and f_i are terms. For a constant a, we write a instead of $a()$. We denote terms using the letters s, t. Syntactic equality of terms s, t is denoted as $s \equiv t$. The set of variables occurring in the term s is denoted as $Var(s)$. A term s is called a *first-order term* if it has no occurrences of context variables and called *ground term* if s has no occurrences of variables.
Contexts are formed using the grammar

$$C[\cdot] ::= [\cdot] \mid X(C[\cdot]) \mid f(t_1, \ldots, \underbrace{C[\cdot]}_{k}, \ldots, t_{ar(f)}),$$

where $[\cdot]$ is called the *hole* (also *trivial context*, *Id*), f is a function symbol, X is a context variable, C is a context, and t_i are terms. Contexts must contain exactly one occurrence of the hole. We denote contexts as $C[\cdot]$, or as C, if it is not ambiguous, and the subterm $X([\cdot])$ is abbreviated as $X(\cdot)$. The notation $C[t]$ means the term where the term t is plugged into the hole of $C[\cdot]$. We denote syntactic equality of contexts by $\equiv$. A *ground context* is a context without occurrences of variables, i.e., it can be seen as a ground term with a single hole, where a signature with the additional constant $[\cdot]$ is used. The size of terms is the number of occurrences of symbols, and the size of contexts is the number of occurrences of symbols not counting the hole. This may be denoted as $size(s)$.

A *(ground) substitution* is a mapping from terms to ground terms with the following properties. A substitution σ can be represented as $\{x_i \to t_i, X_j \to C_j, i = 1, \ldots, n, j = 1, \ldots, m\}$, where $t_i, i = 1, \ldots, n$ is a ground term and $X_j, j = 1, \ldots, m$ is a ground context. σ operates on terms t by replacing all occurrences of variables x_i by $t_i, i = 1, \ldots, n$ and replacing all occurrences of context variables X_j by $C_j, j = 1, \ldots, m$. The replacement of X by $C[\cdot]$ means to replace all subterm occurrences $X(s)$ by $C[s]$, and the replacement of X by

Id is done by replacing all subterm occurrences $X(s)$ by s. The ground substitution $\sigma = \{x_i \to t_i, X_j \to C_j, i = 1,\ldots,n, j = 1,\ldots,m\}$ has as *domain* the set $\{x_i \mid i = 1,\ldots,n\} \cup \{X_j \mid j = 1,\ldots,m\}$ and as *codomain* the set $\{\sigma(x_i) \mid i = 1,\ldots,n\} \cup \{\sigma(X_j) \mid j = 1,\ldots,m\}$.

We will also use *multi-contexts* $C[\cdot_1,\ldots,\cdot_n]$, which are terms over $\Sigma \cup \{[\cdot_1],\ldots,[\cdot_n]\}$, where $[\cdot_i], i = 1,\ldots,n$ are added as constants to the signature, and where every hole $[\cdot_i]$ occurs exactly once in $C[\cdot_1,\ldots,\cdot_n]$.

Terms and contexts can be seen as labelled trees. The tree addresses are also called *positions*, which are words of positive integers. The expression $t|_p$ $(C|_p)$ denotes the subterm (subcontext) of t (of C) at position p.

If C_1, C_2 are contexts, then we denote the context $C_1[C_2[\cdot]]$ also as C_1C_2. A *prefix* of a context C is a context C_1, such that $C_1C_2 \equiv C$ for some context C_2. For a context $C[\cdot] = f(t_1,\ldots,t_{i-1},C'[\cdot],t_{i+1},\ldots,t_n]$ we define $derail1(C) := \{1,\ldots,i-1,i+1,\ldots,n\}$.

In the following we will use the notation $C[\cdot]^n$, where $C[\cdot]$ is a context and n is an integer. This is defined as $C[\cdot]^1 := C[\cdot]$, $C[\cdot]^{n+1} := C[C[\cdot]^n]$. If we use this notation in a term, it is meant as a meta-notation of a term, not as explicit syntax. I.e. $C^n[t]$ is the term $\underbrace{C[\ldots C}_{n}[t]\ldots]$.

A *context unification problem (CUP)* is a set of equations Γ, denoted as $\{s_1 \doteq t_1,\ldots s_n \doteq t_n\}$. We use $\doteq$ as a symmetric operator, i.e. $s \doteq t \in \Gamma$ iff $t \doteq s \in \Gamma$. An equation of the form $X(s) \doteq Y(t)$ is called *flat* equation. The multiset $term(\Gamma)$ is defined to be $\{s,t \mid (s \doteq t) \in \Gamma\}$. The size of Γ is the sum of the sizes of the terms in $term(\Gamma)$, denoted as $size(\Gamma)$. With $Var(\Gamma)$ we denote the set of variables occurring in Γ, and with $Var_i(\Gamma)$ we denote the set $Var(\Gamma) \cap \mathcal{V}_i$ for $i = 1,2$.

A *unifier* of Γ is a ground substitution σ with domain containing $Var(\Gamma)$, which solves all equations in Γ. I.e., $\sigma(s) = \sigma(t)$ for all $(s \doteq t) \in \Gamma$. A CUP Γ is called *unifiable*, if there is a unifier of Γ.

A unifier σ of Γ is called *minimal* if there is no other unifier σ' of Γ with

$$\sum_{X \in Var_2(\Gamma)} (size(\sigma'(X))) < \sum_{X \in Var_2(\Gamma)} (size(\sigma(X))).$$

A ground substitution σ has *exponent of periodicity* n ([Mak77,SSS98]), iff n is the maximal number, such that there is some context variable X and ground contexts A, B, C, such that $\sigma(X) = AB^nC$.

Proposition 2.1. *([SSS98]) There is a constant c, such that for every unifiable context unification problem Γ and for every minimal unifier σ of Γ its exponent of periodicity is at most $2^{c*size(\Gamma)}$.*

Note that an estimation of the constant c is $c \leq 2.14$ [SSS98], hence there are no hidden (large) constants.

The application of this proposition is that for a unifiable CUP Γ, it is possible to focus on a minimal unifier, which has then also an upper bound on the number n of repetitions A^n of a ground context A. This does not directly lead to a

computation of an upper bound on the size of a unifier, since the size of A is not known.

Definition 2.2. *Let Γ be a CUP. We define* second-order prefixes (SO-prefixes) *for Γ as words in $(\mathcal{V}_2)^*$ for occurrences (of variables) in Γ.*

- *Let s,t be terms. If $s \doteq t$ is in Γ, then s,t have empty SO-prefix.*
- *Let f be a function symbol. If an occurrence of a subterm $f(s_1,\ldots,s_n)$ has SO-prefix w, then the subterm occurrence s_i has SO-prefix w for $i = 1,\ldots,n$.*
- *If an occurrence of a subterm $X(s)$ has SO-prefix w, then the occurrence of X has SO-prefix w, and the occurrence of s has SO-prefix $w \cdot X$.*

If for every context variable X and for all SO-prefixes w_1, w_2 of occurrences of X in Γ, $w_1 = w_2$ holds, and for every first-order variable x and all SO-prefixes w_1, w_2 of occurrences of x in Γ, $w_1 = w_2$ holds, then Γ is called a stratified *context unification problem (SCUP).*
Usually, the CUP Γ is clear, so it is in general unambiguous to speak of the SO-prefix without mentioning Γ.

If Γ is an SCUP, then the SO-prefix of a variable (a context variable) is independent of the occurrence, which means it is unique. Hence we may speak of SO-prefixes of context variables or first-order variables instead of SO-prefixes of occurrences.

Example 2.3. Examples for SCUPs are $\{X(x) \doteq Y(y)\}$, whereas the following two context unification problems are non-stratified: $\{X(X(x)) \doteq Y(y)\}$, $\{X(x) \doteq Y(x)\}$.

Example 2.4. Let us consider the SCUP $\{X(f(a)) \doteq f(X(a))\}$, which is translated from the string-unification problem $xf \doteq fx$. The unifiers are $X \to Id$, $X \to f([\cdot])$, $X \to f(f([\cdot]))$,..., which can be represented as $X \to f^n([\cdot])$. Proposition 2.1 then gives us an upper bound for n, if we are only interested in a minimal unifier.

The following two definitions will be essential ones for the decision algorithm SCU in the following (see section 3).

Definition 2.5. *A set of equations $X_1(s_1) \doteq r_1,\ldots,X_n(s_n) \doteq r_n$ is called a* second-order cycle (SO-cycle)*, if the following holds: X_i occurs in $r_{i-1}, i = 2,\ldots,n$, X_1 occurs in r_n, and at least one such occurrence is not at the top. The* length *of an SO-cycle is the number of equations in it.*

An SO-cycle is called ambiguous*, if either for some $i > 1$ the term r_i contains more than 1 occurrence of X_{i+1}, or r_1 contains more than 1 occurrence of X_n. An SO-cycle is* standardized*, if it is of the form*

$$X_1(s_1) \doteq X_2(t_1),\ldots,X_{n-1}(s_{n-1}) \doteq X_n(t_{n-1}), X_n(s_n) \doteq C[X_1(t_n)],$$

where $C[\cdot]$ is a nontrivial context. I.e. all equations but one in the SO-cycle are flat.

We sometimes represent the terms r_i in an SO-cycle as $C_i[X_{i+1}(t'_i)]$, and $C_n[X_1(t'_n)]$ where $C_i, i = 1,\ldots,n$ is a context.

Note that every SO-cycle of length 1 is standardized.
Note also that for an ambiguous SO-cycle

$$X_1(s_1) \doteq C_1[X_2(t_1)], \ldots, X_n(s_n) \doteq C_n[X_1(t_n)]$$

the representation is ambiguous insofar as there is some C_j that contains the context variable X_{j+1} (or C_n contains X_1, respectively). Hence the j^{th} equation could also be written $X_j[s_j] \doteq C'_j[X_{j+1}(t'_j)]$ for a context $C'_j \neq C_j$.

Lemma 2.6. *Let Γ be an SCUP with an SO-cycle*

$$X_1(s_1) \doteq C_1[X_2(t_1)], \ldots, X_n(s_n) \doteq C_n[X_1(t_n)].$$

Then for all i, the position of the hole of $C_i[\cdot]$ has an empty SO-prefix.

Proof. This follows from stratification. □

Definition 2.7. *Let Γ be an SCUP. Let $\sim$ be the equivalence relation on context variables generated by all pairs $X_1 \sim X_2$ where an equation $X_1(s) \doteq X_2(t)$ is in Γ.*
Let $\succ$ be the transitive closure of the relation generated by all the pairs $X_1 \succ X_2$ where X_1, X_2 have empty SO-prefix and there is an equation $X_1(s) \doteq f(t_1, \ldots, t_n)$ in Γ and X_2 occurs in $f(t_1, \ldots, t_n)$.
Let $\gtrsim$ be the pre-ordering generated by the transitive and reflexive closure of $\succ \cup \sim$.
If there are context variables X, Y with $X \succ Y$ and $Y \gtrsim X$, then we say $\gtrsim$ has a cycle conflict.

If $\gtrsim$ has no cycle conflict, then an equivalence class K (of context variables) of $\sim$ is called a second-order cluster (SO-cluster). *An SO-cluster K is called a* top-SO-cluster, *iff the context variables in K are maximal w.r.t. $\gtrsim$.*
The set $EQ(K)$ is a set of equations $s \doteq t$ in Γ, where a context variable from the SO-cluster K occurs at the top-level of s or t.
A top-SO-cluster K, where all equations in $EQ(K)$ are flat, is called flat top-SO-cluster.

Example 2.8. The SCUP $\Gamma_1 = \{X(x_1) \doteq f(Y(x_2)), Y(x_3) \doteq g(X(x_4))\}$ has as orderings:
$\sim = \{(X,X),(Y,Y)\}$, $\succ = \{(X,Y),(Y,X)\}$, and thus $\gtrsim = \{(X,X),(X,Y),(Y,X),(Y,Y)\}$, We have $X \succ Y$, but $Y \gtrsim X$, hence $\gtrsim$ has a *cycle conflict*.
Let another SCUP be $\Gamma_2 = \{X(x) \doteq Y(y), X(z) \doteq f(u, Z(v)), Z(a) \doteq U(b)\}$. The orderings are $X \sim Y, X \succ Z, Z \sim U$. Γ_2 has $\{X, Y\}$ as top SO-cluster, but it is not flat. The SO-cluster $\{Z, U\}$ is flat, but not a top-SO-cluster. The set $EQ(\{X, Y\})$ is the set of equations $\{X(x) \doteq Y(y), X(z) \doteq f(u, Z(v))\}$.
The SCUP $\Gamma_3 = \{X(g(Z(x_1))) \doteq Y(g(x_2)), X(g(y_1)) \doteq Y(g(y_2)), U_1(z_1) \doteq f(a, U_2(z_2))\}$ has a flat top-SO-cluster $\{X, Y\}$.

Remark 2.9. A top-SO-cluster K of Γ is a set of context variables that is connected by equations $X(s) \doteq Y(t)$, and the context variables from K do only occur at top level in terms of $term(\Gamma)$.

3 An overview of the Stratified Context Unification Algorithm (SCU)

3.1 Structure of SCU

The overall idea of the context unification algorithm SCU is to guess the instantiations of the context variables in a controlled top-down way. The SO-prefix as a syntactic criterion permits to identify levels for this top-down guessing. In the case that an SO-cycle of the form $X_1(.) \doteq C_1[X_2(.)], X_2(.) \doteq C_2[X_3(.)], \ldots, X_n(.) \doteq C_n[X_1(.)]$ is detected (or generated), a series of transformations guarantees the elimination of at least one context variable. If there is no SO-cycle, then a careful guessing reduces SO-clusters and finally eliminates a context variable. Eventually, all context variables are eliminated.

The common terminology in higher-order unification procedures distinguishes equations as rigid-rigid ($f(\ldots) \doteq f(\ldots)$), rigid-flexible ($f(\ldots) \doteq X(\ldots)$), and flexible-flexible ($X(\ldots) \doteq Y(\ldots)$). The rigid-rigid case will be treated in the decomposition rules, the rigid-flexible case will be treated in SO-cycle elimination and SO-cluster elimination. The treatment of flexible-flexible equations is done by the rules for elimination of flat SO-clusters.

Note that it would be possible to eliminate all the first-order variables at the beginning by nondeterministically replacing them by terms of the form $X(a)$, where a is some constant. However, there is no real gain in clarity, and since there are steps in the algorithm that can better be described using first-order variables, and moreover the termination proof relies among others on the distinction between first-order and context variables, we refrain from eliminating first-order variables in this way.

Definition 3.1. *SCU is the following nondeterministic algorithm:*
Given an initial stratified context unification problem Γ_{init}, an upper bound E is fixed for the exponent of periodicity (see Proposition 2.1). Then the rules for decomposition (see section 4) are applied exhaustively. The rules for eliminating SO-cycles and SO-clusters are applied (see subsections 5 and 6), where the SO-cycle rules are applied if there is an SO-cycle, and the SO-cluster rules are applied if there is no SO-cycle. After every rule application, there is a subsequent exhaustive application of decomposition rules. This is done until a Fail occurs, or the resulting system is empty. In the latter case, the answer is "yes: unifiable". Using SCU as a decision algorithm is as follows: If there is a possibility to answer "yes: unifiable", then Γ_{init} is recognized as unifiable. If all possibilities end in a Fail, then Γ_{init} is not unifiable

The algorithm SCU computes only a yes/no answer, however, a slight extension would enable it to output a unifier: Following the rule applications in the backwards direction, it is easy to construct a unifier.

In the following we assume that an upper bound E given in Proposition 2.1 for the exponent of periodicity of a unifier of Γ_{init} is fixed, in order to simplify the presentation of the rules.

The (non-deterministic) algorithm is presented by describing rules that are applied to a SCUP Γ, and may output a SCUP Γ', where the number of choices for the output problem is always finite. It is necessary that all rules are effective, and also that all the choices can effectively be computed.

Definition 3.2. *A rule is called* sound, *if whenever the rule transforms an SCUP Γ into the SCUP Γ', unifiability of Γ' implies the unifiability of Γ.*

A rule is called complete, *if for all input SCUPs Γ, and all unifiers σ of Γ with exponent of periodicity $\leq E$, the rule has a possibility to output an SCUP Γ' that has a unifier σ' with exponent of periodicity $\leq E$.*

We show under which circumstances we can claim SCU to be a decision algorithm for unifiability of SCUPs by computing the hole tree of the non-deterministic transformations.

Proposition 3.3. *Assume the following holds:*

1. *Every execution possibility of SCU terminates, if the input is a SCUP.*
2. *Every rule of SCU is sound and complete, and also effective.*
3. *Every rule has only a finite number of execution possibilities.*
4. *SCU stops only in two cases: either it is saying Fail, or the final SCUP is empty.*

Then SCU is a decision algorithm for stratified context unification.

Proof. Since every execution possibility of SCU terminates, and since every rule has only a finite number of choices, using König's Lemma, the computation tree of all possibilities of SCU is finite. Hence it is effectively possible in finite time to compute all possible outputs of SCU and check them whether there is or is not an empty SCUP as output.

If the input SCUP Γ_{init} is not unifiable, then soundness implies that there is no execution possibility that leads to a unifiable (i.e. empty) SCUP.

If the input SCUP Γ_{init} is unifiable, then there is a minimal unifier that has exponent of periodicity $\leq E$ by proposition 2.1. Completeness shows, using induction on the number of rule applications, that there is a final SCUP with a unifier that has also exponent of periodicity $\leq E$. Since the final SCUP is empty, it is unifiable. □

Note that it is possible that there is an execution path of SCU, where the minimal unifier is lost, e.g. for a Γ' on this path there are only unifiers with exponent of periodicity strictly greater than E, but the execution path terminates successfully with an empty SCUP. This does not contradict the method used, since the soundness proof shows that in this case the input SCUP is unifiable, and the completeness part shows, that there is another execution possibility that belongs to a minimal unifier.

Proposition 3.3 structures the correctness proof into showing the following claims:

1. Every execution possibility of SCU terminates, if the input is a SCUP.

2. Every rule of SCU is sound.
3. Every rule of SCU is complete.
4. SCU stops only in two cases: either it is saying Fail, or the final SCUP is empty.

3.2 The Termination Ordering

In order to show termination of SCU, we define a well-founded measure for SCUPs, and show that every rule of SCU strictly decreases this measure.

Definition 3.4. *Let L be an SO-cycle. The measure $\psi(L)$ is a lexicographic combination of the following components:*

1. *The length of L.*
2. *The length of L minus the maximal number of successive flat equations in L.*

Definition 3.5. *The measure μ for termination, also written $\mu(\Gamma)$ is a lexicographic combination $(\mu_1, \ldots, \mu_6)$ of the following well-founded measures:*

1. *μ_1: The number of context variables in Γ.*
2. *μ_2: A measure for SO-cycles: ∞, if there is no SO-cycle, otherwise, the minimal $\psi(L)$ for all SO-cycles in Γ. We use that $\infty > a$ for all $a \neq \infty$.*
3. *μ_3: The number of first-order variables in Γ.*
4. *μ_4: If there is an SO-cycle or if there is no flat top-SO-cluster, then ∞. If there is a flat top-SO-cluster, then the minimal number of context variables in a flat top-SO-cluster.*
5. *μ_5: The multi-set of the sizes of all terms in $term(\Gamma)$ that start with a function symbol. The ordering is the multiset ordering.*
6. *μ_6: The number of equations in Γ.*

Proposition 3.6. *The measure μ is well-founded.*

For more information on orderings, see [DM79,Der87,BN98]

Proposition 3.7. *If for every rule application with input Γ, the output is either Fail, or a Γ' with $\mu(\Gamma) > \mu(\Gamma')$, then every execution of SCU terminates.*

Proof. This follows from the well-foundedness of the measure μ. □

3.3 Stratification

For ensuring termination, it is necessary that all intermediate CUPs are stratified, i.e., are SCUPs.

Proposition 3.8. *If the initial CUP Γ_{init} is stratified, and every rule of SCU keeps stratification, then every intermediate CUP in every run of SCU is also stratified.*

Proof. Follows using induction on the number of rule applications. □

4 Decomposition Rules

4.1 Definition of the Rules

The rules in this subsection transform Γ until all equations are of the form $X(s) \doteq Y(t)$, or $X(t) \doteq f(\ldots)$.

Definition 4.1. *We describe the basic rules, also called the decomposition rules.*

- *(Replace-variable) If $x \doteq t$ is an equation in Γ, and $x \notin Var(t)$, then remove the equation $x \doteq t$, and replace x by t everywhere in Γ.*
- *(Decomposition) Replace an equation $f(s_1, \ldots, s_n) \doteq f(t_1, \ldots, t_n)$ by the equations $s_1 \doteq t_1, \ldots, s_n \doteq t_n$.*
- *(Trivial) Remove equations $s \doteq s$ from Γ.*
- *(Clash) Return Fail, if there is an equation $g(s_1, \ldots, s_m) \doteq f(t_1, \ldots, t_n)$, where f, g are function symbols and $g \neq f$.*
- *(Occurs-check) Return Fail, if there is an equation $x \doteq t$, where $x \in Var(t)$, and $t \not\equiv x$.*

A SCUP Γ is called decomposed, *if no decomposition rule is applicable.*

Note that the rule $X(s) \doteq X(t) \rightarrow s \doteq t$ is not used, since it would destroy stratification. It is used for several equations at once in a rule solving SO-clusters.

4.2 Correctness of Decomposition Rules

Lemma 4.2. *Every decomposition rules either fails, or transform an input SCUP into a stratified CUP.*

Proof. The critical operation is replacing a first order variable. Replacing a first-order variable x by t triggered by an equation $x \doteq t$ in Γ retains stratification, since the replacement is done everywhere, and the SO-prefix of x is empty at every occurrence. Thus the SO-prefixes of the new occurrences of variables in t remain the same after the replacement. □

Lemma 4.3. *The decomposition rules are sound and complete*

Proof. The proof for all rules except occurs-check is the standard one for decomposition rules in the first-order case.

For the rule (occurs-check), it is obvious that $x \doteq f(t_1, \ldots, t_n)$ is not unifiable if x occurs in some t_i. For an equation $x \doteq X(t)$, stratification ensures that x is not contained in t. □

Note that the situation is the same as for first order unification problems. Basically, the transformations do not change the set of unifiers.

Lemma 4.4. *The decomposition rules strictly decrease the measure μ:*

Proof. The rule (Replace-variable) either generates an SO-cycle, i.e. it may strictly reduce μ_2, or it leaves this invariant, and strictly decreases the number of first-order variables. The rule (Decomposition) may generate an SO-cycle, or a flat top-SO-cluster, otherwise it strictly decreases μ_5. The rule (Trivial) strictly reduces μ_6. □

4.3 Property of Decomposed CUPs

Proposition 4.5. *Let Γ be a nonempty and decomposed SCUP. Then either Γ has an SO-cycle, or there is a top-SO-cluster.*

Proof. This follows by standard arguments on the orderings in definition 2.7. If there is a cycle-conflict, then no SO-cluster is defined, however, the generation of the ordering from the basic relations shows that in this case an SO-cycle is in Γ. □

5 Elimination of SO-cycles

5.1 Definition of the Rules

In this subsection we describe transformation rules that operate on SO-cycles. Assume we have a shortest SO-cycle:

$$X_1(.) \doteq C_1[X_2(.)], X_2(.) \doteq C_2[X_3(.)], \ldots, X_n(.) \doteq C_n[X_1(.)]$$

The critical path of the transformation is as follows: First contexts C_i are shifted in a shortest SO-cycle to a single equation, i.e., such that a standardized SO-cycle of the form

$$X_1(s_1) \doteq X_2(t_1), \ldots, X_{h-1}(s_{h-1}) \doteq X_h(t_{h-1}), X_h(s_h) \doteq C_h[X_1(t_h)]$$

is generated. The next step is to operate on this standardized SO-cycle.
Two cases have to be distinguished: i) that X_1 occurs in C_h, which due to the stratification condition can only be with trivial SO-prefix and ii) that X_i does not occur in C_h. In any case the SO-cycle will be shortened by some transformation. The last step is to use the bound E on the exponent of periodicity to eliminate one context variable in an SO-cycle of length 1.

The non-critical possibilities of the algorithm are for example eliminating a context variable, or generating a (different) shorter SO-cycle, which may then be used as the target of the transformations, since then the measure μ is strictly decreased.

The possibilities of the rules are either that a context variable can be instantiated (CV-eliminate, Partial-prefix), or there is a standardization operation (Full-prefix), or the position of the hole of some context variable does not follow the direction (rails) given by the selected SO-cycle. We call this possibility "derailing". After a derailing, the operated-upon SO-cycle is destroyed, and after that the remaining pieces contain a shorter SO-cycle.
We make the following assumption on the applicability of the rules in this section:

- The SO-cycle elimination rules are only applied if the SCUP is decomposed.
- The SO-cycle elimination rules are applied to a shortest SO-cycle.

Definition 5.1. *Rule (Standardize-cycle)*
This rule is only applicable, if a shortest SO-cycle is non-standardized. Let L be such an SO-cycle in Γ that has among the shortest, non-standardized SO-cycles the maximal number of successive flat equations. In particular, the SO-cycle has length at least 2.
The SO-cycle L can be represented as

$$X_1(s_1) \doteq X_2(t_1), \ldots, X_{j-1}(s_{j-1}) \doteq X_j(t_{j-1}),$$
$$X_j(s_j) \doteq C_j[X_{j+1}(t_j)], \ldots, X_h(s_h) \doteq C_h[X_1(t_h)]$$

All equations with index $i < j$ are flat, and the equation with index j is non-flat, i.e., C_j, C_h are nontrivial and $j \neq h$. Note that there is no occurrence of $X_i, i = 1, \ldots, j$ in C_j.
Then select one of the following possibilities:

1. *(CV-elimination) Select some $i \in [1..h]$ and instantiate X_i by Id, i.e., remove it from Γ.*
2. *(Partial-prefix) Select some prefix $C_{j,1}$ of C_j, i.e., $C_j \equiv C_{j,1}C_{j,2}$, and replace X_i either by $C_{j,1}[X'_i(\cdot)]$ or by $C_{j,1}[\cdot]$ for all $i = 1, \ldots, j$, where X'_i are new context variables. For at least one $i = 1, \ldots, j$, select the replacement of X_i by $C_{j,1}[\cdot]$.*
3. *(Full Prefix) For all $i = 1, \ldots, j$, replace X_i using $X_i = C_j[X'_i(\cdot)]$ where X'_i are new context variables. Apply (Decomposition) to the first j equations after the instantiation until (among others) the equations $X'_i(s_i) \doteq X'_{i+1}(t_i)$ for $i = 1, \ldots, j-1$ and $X'_j(s_j) \doteq X_{j+1}(t_j)$ are obtained.*
4. *(Derailing) Select a prefix $C_{j,1}$ of C_j, such that $C_j \equiv C_{j,1}C_{j,2}$, and $C_{j,2}$ is not trivial. Let f be the top level function symbol of $C_{j,2}$ of arity n. Fail, if $n = 1$.*
   For every $i = 1, \ldots, j$, select an index $1 \leq k_i \leq n$, such that for at least one i: $k_i \in derail1(C_{j,2})$. Replace X_i by $C_{j,1}[f(x_{i,1}, \ldots, \underbrace{X'_i(\cdot)}_{k_i}, \ldots, x_{i,n})]$, where X'_i are new context variables and $x_{i,k}$ are new variables.
   Then exhaustively apply (Decomposition) and after that exhaustively apply (Replace-variable) to the equations that are the result of instantiating the first j equations of the SO-cycle.

If several successive applications of the rule (Standardize-cycle) do not strictly reduce the number μ_1 of context variables or the length of a shortest SO-cycle, then it will transform in several steps a minimal SO-cycle into a (minimal) standardized SO-cycle.

Example 5.2. An illustrating example for an application of the rule (Standardize-cycle) is the SCUP

$$\{X_1(x_1) \doteq f(X_2(y_1)), X_2(x_2) \doteq g(X_3(y_2), Y(a)), X_3(x_3) \doteq h(X_1(y_3))\}$$

Note that this is a non-ambiguous SO-cycle.

The assumption that the rule is applied only to shortest SO-cycles prevents occurrences of e.g. X_1 in the right term of the second equation. Applying the rule (Standardize-cycle) to it using the third possibility using the replacement

$$X_1 := f(X_1'(\cdot))$$

results after decomposition in

$$\{X_1'(x_1) \doteq X_2(y_1), X_2(x_2) \doteq g(X_3(y_2), Y(a)), X_3(x_3) \doteq h(f(X_1'(y_3)))\}$$

Now there is one flat equation in the SO-cycle.
We show two different possibilities:

1. We may choose possibility 3 of the rule.
   Let the replacements be

   $$X_1' := g(X_1''(\cdot), Y(a)), X_2 := g(X_2''(\cdot), Y(a))$$

   The result is:

   $$\{X_1''(x_1) \doteq X_2''(y_1), X_2''(x_2) \doteq X_3(y_2), \\ X_3(x_3) \doteq h(f(g(X_1''(y_3), Y(a))))\}$$

   Now the SO-cycle is standardized.
2. We may also choose possibility 4 (derailing).
   Applying the replacements

   $$X_1' := g(X_1''(\cdot), z_1), X_2 := g(z_2, X_2''(\cdot))$$

   to

   $$\{X_1'(x_1) \doteq X_2(y_1), X_2(x_2) \doteq g(X_3(y_2), Y(a)), X_3(x_3) \doteq h(f(X_1'(y_3)))\}$$

   results in:

   $$\{X_1''(x_1) \doteq z_2, z_1 \doteq X_2''(y_1), \\ z_2 \doteq X_3(y_2), X_2''(x_2) \doteq Y(a), \\ X_3(x_3) \doteq h(f(g(X_1''(y_3), z_1)))\}$$

   Application of the rule (Replace-variable) for the variable z_2 generates the following SO-cycle of length 2:

   $$X_1''(x_1) \doteq X_3(y_2), X_3(x_3) \doteq h(f(g(X_1''(y_3), z_1)))$$

Example 5.3. A slight variation shows the application of (Standardize-cycle) to a SCUP with an ambiguous SO-cycle.

$$\{X_1(x_1) \doteq f(X_2(y_1)), X_2(x_2) \doteq g(X_3(y_2), Y(a), X_3(a)), X_3(x_3) \doteq h(X_1(y_3))\}$$

Applying the rule (Standardize-cycle) to it using the third possibility using the replacement

$$X_1 := f(X_1'(\cdot))$$

results after decomposition in

$$\{X_1'(x_1) \doteq X_2(y_1), X_2(x_2) \doteq g(X_3(y_2), Y(a), X_3(a)), X_3(x_3) \doteq h(f(X_1'(y_3)))\}$$

Now we choose possibility 3. Let the replacements be

$$X_1' := g(X_1''(\cdot), Y(a), X_3(a)), X_2 := g(X_2''(\cdot), Y(a), X_3(a))$$

The result after decomposition is

$$\{X_1''(x_1) \doteq X_2''(y_1), X_2''(x_2) \doteq X_3(y_2),$$
$$X_3(x_3) \doteq h(f(g(X_1''(y_3), Y(a), X_3(a))))\}$$

Analyzing the result, there is now the SO-cycle

$$X_3(x_3) \doteq h(f(g(X_1''(y_3), Y(a), X_3(a))))$$

of length 1.

Now we describe two rules that operate on standardized SO-cycles, either removing a context variable, or shortening the SO-cycle. The difference lies in the derailing part. If the SO-cycle is ambiguous, then only one instantiation round in the SO-cycle is necessary. In a standardized SO-cycle, there may be more instantiation rounds before derailing. The number of rounds can be limited by the exponent of periodicity.

Definition 5.4. *Rule (Solve-standardized-ambig-cycle)*
This rule has to be applied to a shortest SO-cycle that is in addition standardized and ambiguous: Let the SO-cycle be of the form

$$X_1(s_1) \doteq X_2(t_1), \ldots, X_{h-1}(s_{h-1}) \doteq X_h(t_{h-1}), X_h(s_h) \doteq C[X_1(t_h)],$$

where C is a context, and X_1 occurs in C.
Select one of the following possibilities:

1. *(CV-elimination) Select some $X_i, i \in [1..h]$ and instantiate it by Id, i.e., remove it from Γ.*
2. *(Partial-prefix) Select C_1, C_2 such that $C \equiv C_1C_2$, C_1, C_2 are not trivial, and X_1 does not occur in C_1. Fail, if this is not possible.*
   Replace X_i either by $C_1[X_i'(\cdot)]$ or by $C_1[\cdot]$ for all $i = 1, \ldots, h$, where X_i' are new. There must be at least one index $j \in [1..h]$ such that X_j is replaced by $C_1[\cdot]$.
3. *(Derailing) This case is only applicable, if the SO-cycle has length $h > 1$. Select C_1, C_2 such that $C \equiv C_1C_2$, C_2 is not trivial, and X_1 does not occur in C_1. Fail, if this is not possible.*
   Let f be the top level function symbol of C_2 and let $n = ar(f)$. Fail, if $n = 1$. For every i, select an index $1 \leq k_i \leq n$, such that for at least one i, $k_i \in derail1(C_2)$.

Replace X_i *by* $C_1[f(x_{i,1},\ldots,\underbrace{X'_i(\cdot)}_{k_i},\ldots,x_{i,n})]$ *for* $i=1,\ldots,h$, *where* $x_{i,k}, X'_i$ *are new.*
Then exhaustively apply (Decomposition) and after that exhaustively apply (Replace-variable) to the equations that are the result of instantiating the equations in the SO-cycle.

Definition 5.5. *Rule (Solve-standardized-cycle)*
This rule has to be applied to a shortest SO-cycle that is in addition standardized and not ambiguous. Let the SO-cycle be of the form

$$X_1(s_1) \doteq X_2(t_1),\ldots,X_{h-1}(s_{h-1}) \doteq X_h(t_{h-1}), X_h(s_h) \doteq C[X_1(t_h)],$$

where C *is a context, and* X_1 *is not contained in* C.
Let $0 \le e \le E$ *be an integer where* E *is the fixed upper bound for the exponent of periodicity, given in Proposition 2.1 for the initial problem* Γ_{init}.
Select one of the following possibilities:

1. *(CV-elimination) Select some* $X_i, i \in [1..h]$ *and instantiate it by* Id, *i.e., remove it from* Γ.
2. *(Partial-prefix) Let* C_1 *be a prefix of* C. *Replace* X_i *either by* $C^e[C_1[X'_i(\cdot)]]$ *or by* $C^e[C_1[\cdot]]$ *for all* $i=1,\ldots,h$, *where* X'_i *is a new context variable.*
   For at least one $i=1,\ldots,h$, *select the replacement of* X_i *by* $C^e[C_1[\cdot]]$.
3. *(Derailing) This selection is only possible for* $h>1$. *Let* C_1 *be a prefix of* C, *such that* $C_1C_2 \equiv C$ *and* C_2 *is nontrivial. Let* f *be the top level function symbol of* C_2 *and let* $n := ar(f)$. *Fail, if* $n=1$.
   For every i, *select an index* $1 \le k_i \le n$, *such that for at least one* i, $k_i \in derail1(C_2)$.
   For all $i=1,\ldots,h$, *replace* X_i *by* $C^eC_1[f(x_{i,1},\ldots,\underbrace{X'_i(\cdot)}_{k_i},\ldots,x_{i,n})]$, *where* $x_{i,k}, X'_i$ *are new.*
   Then exhaustively apply (Decomposition) and after that exhaustively apply (Replace-variable) to the equations that are the result of instantiating the equations in the SO-cycle.

Example 5.6. Consider the SCUP $\{X(x) \doteq g(f(X(a),X(a)),b)\}$. There are infinitely many solutions as second-order unification problem, but as a SCUP, there are only finitely many possibilities for X. The possibilities are covered in the rule (Solve-standardized-ambig-cycle). One is $X=Id$, which results in the unifier $\{x \doteq g(f(a,a),b)\}$. The other possibility is $X = g(X'(\cdot),b)$. The SCUP after applying the instantiation $X = g(X'(\cdot),b)$ is:

$$X'(x) \doteq f(g(X'(a),b),g(X'(a),b))$$

Now only $X'=Id$ is possible, which results in a unifiable SCUP.
Suppose the instantiation were $X' = f(X''(\cdot),t)$. Then $t \doteq g(f(X''(a),t),b)$ has to be unifiable, which is not possible since the sizes are different. The same

holds for the instantiation $X' = f(t, X''(\cdot))$. These instantiations are precluded in the rule (Solve-standardized-ambig-cycle), since (Derailing) is not possible for an SO-cycle of length 1, and the selection (Partial-prefix) is not possible, since every nontrivial prefix C_1 would contain the context variable X'.

5.2 Correctness of SO-Cycle-Elimination

Lemma 5.7. *The SO-cycle rules either fail, or transform an input SCUP into a stratified CUP.*

Proof.

- Intermediate decomposition keeps stratification, as already proved in lemma 4.2
- Removing a context variable $X(\cdot)$ keeps stratification, since it is removed from all SO-prefixes.
- Replacing a context variable X with empty SO-prefix by a context, where the hole has empty SO-prefix also keeps stratification. That this is the only possibility is stated in lemma 2.6.
- Replacing a context variable X by $C[X'(\cdot)]$ is done only if the SO-prefix is empty, and the new SO-prefix for X' is also empty. The variable X in SO-prefixes is replaced by X', hence the stratification condition for other variables remains true.

□

Lemma 5.8. *The rules for SO-cycle elimination are sound and complete.*

Proof. Soundness can easily be verified by inspecting the rules.

Completeness: Let σ be a (ground) unifier of Γ with an exponent of periodicity $\leq E$. In the proof below it is shown that an output SCUP Γ' exists. The construction of a corresponding unifier σ' is either obvious, or there are hints on the construction. For the SO-cycle elimination rules the following is easy to verify: The (ground) unifier σ' has also an exponent of periodicity not greater than E, since every ground context in the codomain of σ' is already a subcontext of a term or context in the codomain of σ.

We show which Γ' may be selected to show completeness.
If there is some X_i in the SO-cycle with $\sigma(X_i) = Id$, then we use selection 1) in the appropriate rule. Hence in the rest of this proof we can assume that $\sigma(X_i) \neq Id$ for all i.
The cases are:

1. The (minimal) SO-cycle is non-standardized. Then $h > 1$. The rule (Standardize-cycle) will be applied. Let A be the greatest common prefix of the ground instances $\sigma(X_i)$ of the context variables X_i for $i = 1, \ldots, j$ and of $\sigma(C_j)$.
   The different cases are:

- $\sigma(X_i) \equiv A$ for some i. Then selection 2 is used.
- $\sigma(C_j) \equiv A$. Then selection 3 is used.
- A is a proper prefix of $\sigma(C_j)$ and of all $\sigma(X_i), i = 1, \ldots, j$. Let A_i be such that $\sigma(X_i) = AA_i$, $i = 1, \ldots, j$ and let $\sigma(C_j) = AC'_j$. The context C'_j is nontrivial. Since σ is a unifier, the top level function symbol f of all A_i and C'_j is the same. C_1, C_2 can be select such that $\sigma(C_{j,1}) = A, \sigma(C_{j,2}) = C'_j$. Since A is a proper prefix of $\sigma(X_i)$ and $\sigma(C_j)$, the arity of f is greater than 1. For $i = 1, \ldots, j$, the index k_i is chosen as the first character of the position of the hole of A_i. There must be one k_i in $derail1(\sigma(C_{j,2}))$, since otherwise the prefix A is not maximal.

In every case we can choose a possibility in (Standardize-cycle), such that the output SCUP has a unifier.

2. The SO-cycle is standardized and ambiguous. In this case the rule (Solve-standardized-ambig-cycle) is used.

   First we treat the case that the SO-cycle has length 1. Then the SO-cycle can be represented as $X_1(s_1) \doteq f(t_1, \ldots, t_n)$. Assume, there are two indices $k \neq j$, such that X_1 occurs in t_k and t_j. W.l.o.g. assume that $k \in derail1(\sigma(X_1))$. Then $\sigma(t_j)$ is properly contained in $\sigma(X_1)$, which contradicts the fact that $\sigma(X_1)$ is contained in $\sigma(t_j)$. Thus there is only one index j, such that t_j contains occurrences of X_1. By induction on the length of the largest common prefix of the paths to all occurrences of X_1 in C, there is a nontrivial context C_1 with $C \equiv C_1C_2$, and without occurrences of X_1, such that $\sigma(X_1) = \sigma(C_1)$. We can use selection 2.

   For the case $h > 1$ let A be the greatest common prefix of $\sigma(X_i), i = 1, \ldots, h$ and of $\sigma(C)$. Assume $A = \sigma(C)$. Since X_1 is contained in C, this implies that A as a prefix of $\sigma(X_1)$ would be properly contained in $\sigma(C) = A$, which is a contradiction.

   Hence A is a proper prefix of $\sigma(C)$, and we can select C_1, C_2, such that $C = C_1C_2$, C_2 is not trivial, and $\sigma(C_1) = A$. It is clear that X_1 cannot occur in C_1. If $\sigma(X_i) = A$ for some i, then use selection 2. Otherwise use selection 3. Now we can use similar arguments as for the rule (Standardize-cycle).

3. The SO-cycle is standardized and not ambiguous. In this case the rule (Solve-standardized-cycle) is used.

   Let A be the greatest common prefix of $\sigma(X_i)$ for $i = 1, \ldots h$ and of $\sigma(C^E)$. Let $A = \sigma(C)^e C'$ with $0 \leq e \leq E$ and C' a proper prefix of $\sigma(C)$, let C'' be a context such that $C'C'' = \sigma(C)$. This choice of e is possible, since we have assumed that the exponent of periodicity of σ is not greater than E.

   First let $h = 1$. We have to show that $\sigma(X_1) = A$. Assume this is false. Then $\sigma(X_1) = Af(t_1, \ldots, \underbrace{\cdot}_{k}, \ldots, t_n)D$ for some ground context D, where $Af(t_1, \ldots, \underbrace{\cdot}_{k}, \ldots, t_n)$ is not a prefix of $\sigma(C^E)$. Since σ solves the equations in the SO-cycle, the equation

$$Af(t_1, \ldots, \underbrace{t'}_{k}, \ldots, t_n) = \sigma(C)\, Af(t_1, \ldots, \underbrace{t''}_{k}, \ldots, t_n)$$

holds. Decomposition shows:

$$f(t_1, \ldots, \underbrace{t'}_{k}, \ldots, t_n) = C''f(t_1, \ldots, \underbrace{t''}_{k}, \ldots, t_n)$$

Let k' be the first letter of the position of the hole of C''. Then $k' \neq k$ by assumption, and $t_{k'}$ contains $f(t_1, \ldots, t_{k'}, \ldots, \underbrace{t''}_{k} \ldots, t_n)$, which is a contradiction. Now it is clear that selection 2 can be used.

Now let $h > 1$. If $\sigma(X_i) = A$ for some i, then use selection 2. If A is a proper prefix of all $\sigma(X_i)$, then let f be the top level function symbol of C''. This function symbol must be of arity at least 2, since otherwise the common prefix A would be longer, for σ solves the equations in the SO-cycle. Select the prefix C_1 of C such that $\sigma(C_1) = C'$. Now we select the indices k_i as the first letter of the position of the hole of A_i, where $\sigma(X_i) = AA_i$ for i = 1,...,h. It is also clear that there is some i, such that k_i is not the first letter of the position of the hole in $C'' = \sigma(C_2)$, since otherwise, the prefix A would not be maximal.

□

5.3 Termination of SO-Cycle Rules

We show in this subsection, that every application of an SO-cycle rule either decreases the measure μ, or leads to a Fail.

Lemma 5.9. *The rule (Standardize-cycle) strictly decreases the measure μ.*

Proof. First we argue that the instantiations do not increase the number of context variables. Assume there is a shortest non-standardized SO-cycle. Since the SO-cycle is a shortest one, the context C_j does not contain $X_i, i = 1, \ldots, j$, since $j < h$, and thus the instantiations do not contain the context variables $X_i, i = 1, \ldots, j$.

The rule (Standardize-cycle) either strictly reduces the number of context variables (possibilities 1 and 2), or generates a new SO-cycle of shorter length than h, or of the same length, but a strictly greater number of successive flat equations (possibility 3). Hence μ is strictly decreased for selections 1,2, and 3.

The selection possibility 4 strictly decreases the length of a shortest SO-cycle after applying exhaustively (Decomposition) and (Replace-variable), as can be seen as follows:

Let the context $C_{j,3}$ and the index k_{j+1} be defined by $C_{j,2} = f(r_1, \ldots, \underbrace{C_{j,3}[\cdot]}_{k_{j+1}}, \ldots, r_n)$.

After instantiating the first j equations and exhaustively applying (Decomposition) (i.e., decomposing away $C_{j,1}$), the following first j equations are obtained:

$$
\begin{aligned}
f(x_{1,1},\ldots,\underbrace{X'_1(s_1)}_{k_1},\ldots,x_{1,n}) &\doteq f(x_{2,1},\ldots,\underbrace{X'_2(t_1)}_{k_2},\ldots,x_{2,n})\\
&\ldots\\
f(x_{j-1,1},\ldots,\underbrace{X'_{j-1}(s_{j-1})}_{k_{j-1}},\ldots,x_{j-1,n}) &\doteq f(x_{j,1},\ldots,\underbrace{X'_j(t_{j-1})}_{k_j},\ldots,x_{j,n})\\
f(x_{j,1},\ldots,\underbrace{X'_j(s_j)}_{k_j},\ldots,x_{j,n}) &\doteq f(r_1,\ldots,\underbrace{C_{j,3}[X_{j+1}(t_{j-1})]}_{k_{j+1}},\ldots,r_n)
\end{aligned}
$$

Applying (Decomposition) and focussing the position k_{j+1} yields the following equations. For index j in the SO-cycle the equation is one of

$$
\begin{aligned}
x_{j,k_{j+1}} &\doteq C_{j,3}[X_{j+1}(t_{j-1})] \quad \text{for } k_j \neq k_{j+1}\\
X'_j(s_j) &\doteq C_{j,3}[X_{j+1}(t_{j-1})] \quad \text{for } k_j = k_{j+1}
\end{aligned}
$$

For the other equations, the following pairs are possible:

$$
\begin{aligned}
u_{i-1} &\doteq x_{i,k_{j+1}}\\
x_{i,k_{j+1}} &\doteq u_i
\end{aligned}
$$

or

$$
\begin{aligned}
u_{i-1} &\doteq X'_i(t_{i-1})\\
X'_i(s_i) &\doteq u_i
\end{aligned}
$$

Since k_{j+1} is different from at least one k_i with $1 \leq i \leq j$, the chain of equations contains at most j context variables at the top, i.e., at least one is missing. Moreover, after replacing variables $x_{i,j}$, a strictly shorter SO-cycle is obtained, since both $x_{1,k_{j+1}}$ and X'_1 is contained in the right term of the equation with index h of the input SO-cycle L after the first instantiation. It is also clear that all the freshly introduced variables $x_{i,j}$ will be replaced, and thus the measure μ_3 is not increased.
In summary, the measure μ is strictly decreased.

Note that an occurs-check situation (as in the next proofs) is not possible, since the SO-cycle is not standardized, and $C_{j,2}$ is not trivial. □

Lemma 5.10. *After application of the rule (Solve-standardized-ambig-cycle) either the measure μ is strictly decreased or there is an occurs-check failure.*

Proof. First it is clear that C_1 in the definition of the rule does not contain any of the context variables X_i due to the following reasons: the SO-cycle is of minimal length, we assumed that X_1 is not contained in C_1, and Γ is stratified.

The selection possibilities 1 and 2 of rule (Solve-standardized-ambig-cycle) strictly reduce the number of context variables.

If selection 3 is applied, then it generates a new SO-cycle of shorter length than h after decomposition:
The arguments are almost the same as for the possibility 4 of the termination proof of (Standardize-cycle) of Lemma 5.9 to show that the length of the SO-cycle is strictly reduced and that the number of first order variables is not increased. There is exactly one exception: if after decomposition there are only variable equations, i.e., if $k_j \neq k_{h+1}$ for all j, where k_{h+1} is the first letter of the position of the hole in C_2. Let C_3, r_i be determined by $C_2 = f(r_1, \ldots, \underbrace{C_3[\cdot]}_{k_{h+1}}, \ldots, r_n)$, and assume that all k_j are different from k_{h+1}. Then after decomposition, the equations are of the form

$$\begin{aligned} x_{1,k_{h+1}} &\doteq x_{2,k_{h+1}} \\ &\ldots \\ x_{h-1,k_{h+1}} &\doteq x_{h,k_{h+1}} \\ x_{h,k_{h+1}} &\doteq C_3C_1[f(x_{1,1}, \ldots, x_{1,k_{h+1}}, \ldots, \underbrace{X_1'(t_j)}_{k_1}, \ldots, x_{1,n})] \end{aligned}$$

and after some applications of (Replace-variable), the occurs-check failure rule is applicable. □

Lemma 5.11. *After application of the rule (Solve-standardized-cycle) either the measure μ is strictly decreased or there is an occurs-check failure:*

Proof. Since the SO-cycle is not ambiguous, cases 1 and 2 strictly decrease the measure μ.

In the derailing case, after instantiating and decomposing the equations and looking for the equations that result from eliminating the context C^eC_1 by decomposition, the same arguments as in the proof of termination of (Solve-standardized-ambig-cycle) can be used (see proof of Lemma 5.10). □

6 Elimination of SO-Clusters

6.1 Definition of the SO-Cluster Elimination Rules

This subsection contains rules to resolve top-SO-clusters K. The critical path of the transformation is first to generate a flat top-SO-cluster from non-flat ones by a generalized imitation rule. Once a flat top-SO-cluster is generated, a guess by a generalized flexible-flexible rule generates smaller and smaller top-SO-clusters, until it is possible to remove a context variable. The uncritical paths

of the transformation are that a context variable is eliminated or an SO-cycle is generated, which strictly reduces the measure of Γ.
We assume that the rules in this subsection are only applied to decomposed SCUPs.

Definition 6.1. *Rule (Solve-non-flat-cluster)*
This rule is only applicable if there are no SO-cycles, no flat top-SO-clusters, but a non-flat top-SO-cluster.

Let $K = \{X_1, \ldots, X_h\}$ be a non-flat top-SO-cluster, and let $X_j(s) \doteq f(t_1, \ldots, t_n)$ be an equation in $EQ(K)$.
Then select one of the following two possibilities:

1. *(CV-elimination) Select some $X_i, i \in [1..h]$ and instantiate it by Id, i.e., remove it from Γ.*
2. *(Rigid-flexible) For every $i = 1, \ldots, h$, select an index $1 \leq k_i \leq n$ and replace every $X_i \in K$ by $f(x_{i,1}, \ldots, \underbrace{X'_i(\cdot)}_{k_i}, \ldots, x_{i,n})$ where $x_{i,j}, X'_i$ are new variables.*

   Then one application of (Decomposition) has to be made for every resulting equation from $EQ(K)$ and afterwards use (Replace-variable) for the variables $x_{i,j}$.

In the following rule we use a new function symbol F, which makes it easier to describe the effects of the rule, and permits also an infinite signature Σ. This function symbol is only for intermediate use, since it is not contained in the resulting SCUP.

Definition 6.2. *Rule (Solve-flat-cluster)*
This rule is applicable if there are no SO-cycles, but a flat top-SO-cluster.
Let $K = \{X_1, \ldots, X_h\}$ be a flat top-SO-cluster of minimal cardinality.
Select one of the following possibilities:

1. *(CV-elimination) Select some $X_i, i \in [1, , h]$ and instantiate it by Id, i.e., remove it from Γ.*
2. *(flexible-flexible branching) This case requires $|K| > 1$ and that the maximal arity of function symbols in the signature Σ is greater than 1.*
   Let F be a new function symbol with $n := ar(F)$ with $2 \leq n \leq |K|$.
   For every context variable $X_i \in K$, select an index $1 \leq k_i \leq n$ and replace X_i by $F(x_{i,1}, \ldots, \underbrace{X'_i(\cdot)}_{k_i}, \ldots, x_{i,n})$, where $x_{i,j}, X'_i$ are new. There must be different indices k_i.
   Then exhaustively apply (Decomposition) and after that exhaustively apply (Replace-variable) to the equations that are the result of instantiating the equations in the SO-cluster.

Note that the function symbol F will be eliminated by the rule after decomposing the instantiated equations.

Example 6.3. A (previously non-visible) SO-cycle may pop up after application of rules.
Let the SCUP be $\{X(Z(x)) \doteq X(g(Z(y)))\}$. Then there is no SO-cycle, but a top-SO-cluster $\{X\}$. Guessing $X = Id$ gives the SCUP $\{Z(x) \doteq g(Z(y))\}$, which has an SO-cycle.

Example 6.4. The following SCUP demonstrates the rule for flat SO-clusters. Let Γ be:

$$X(a) \doteq Y(b)$$
$$Y(a) \doteq X(b)$$

The guess $X = Y = Id$ leads to Fail.
Selection 2 of (Solve-flat-cluster) has to be used: A guess may be $X = F(X'(\cdot), x), Y = F(y, Y'(\cdot))$. The resulting SCUP is:

$$F(X'(a), x) \doteq F(y, Y'(b))$$
$$F(y, Y'(a)) \doteq F(X'(b), x)$$

Decomposition and variable-replacement yield $X'(a) \doteq X'(b), Y'(a) \doteq Y'(b)$, which leads to failure after another application of the rule.

If the rule (Solve-flat-cluster) would allow to guess instantiations with equal indices of the holes, then a correct guess would be $X = F(X'(\cdot), x), Y = F(Y'(\cdot), y)$ and the resulting SCUP would be:

$$F(X'(a), x) \doteq F(Y'(b), y)$$
$$F(Y'(a), y) \doteq F(X'(b), x)$$

This leads to the SCUP

$$X'(a) \doteq Y'(b)$$
$$Y'(a) \doteq X'(b)$$

which is a renamed variant of the initial SCUP, and hence termination of the algorithm would be lost.

6.2 Correctness of SO-Cluster Elimination Rules

Lemma 6.5. *The SO-cluster rules either fail or transform an input SCUP into a stratified CUP.*

Proof. We check the different possible instantiations:

- Intermediate decomposition keeps stratification, as already proved in lemma 4.2.

- Removing a context variable $X(\cdot)$ keeps stratification, since it is removed from all SO-prefixes.
- Replacing a context variable X with empty SO-prefix by a context of the form $F(x_{i,1}, \ldots, X'(\cdot), \ldots, x_{i,n})$ keeps stratification, since all occurrences of X' have empty SO-prefix, and in all other SO-prefixes, the X is replaced by X'.

□

Lemma 6.6. *The rules for SO-cluster elimination are sound.*

Proof. We check the cases:

- The rule (Solve-non-flat-cluster) is applied to a non-flat top-SO-cluster K. This is sound, since the rule applies only instantiations.
- The rule (Solve-flat-cluster) was applied to the SO-cluster K. There are the following cases:
  1. $h = 1$.
     Then all the equations in $EQ(K)$ are of the form $X_1(s_i) \doteq X_1(t_i)$. The transformation replaces these equations by the equations $s_i \doteq t_i$. This is sound, since a unifier of the resulting Γ' can be modified to a unifier of Γ by using an arbitrary instantiation for X_1.
  2. The signature contains only constants and unary function symbols.
     Then only guessing some X_j as Id can be used, which is sound.
  3. The signature contains some function symbol of arity at least two, and $|K| > 1$.
     We show soundness of the second selection possibility. Let σ' be a unifier after the application of the rule. The interesting part is to construct an instantiation for the variables $X_i, i = 1, \ldots, h$ before application. It is easy to see that the application is sound in a signature extended with the function symbol F. This leads to a unifier σ before application of the rule, where $\sigma(X_i)$ has a top occurrence of F. Instead of F, use a ground multi-context $t[\cdot_1, \ldots, \cdot_n]$ with n holes, where n is the arity of F, and t has no occurrence of F. Such a multi-context can be constructed, if there is a function symbol of arity at least 2. Construct a unifier σ'' of the input SCUP by replacing every term $F(t_1, \ldots, t_m)$ by $t[t_1, \ldots, t_m]$.

□

Lemma 6.7. *The rules for SO-cluster elimination are complete.*

Proof. Let σ be a unifier of Γ with an exponent of periodicity $\leq E$. In the proof below it is shown that an output SCUP Γ' exists. The construction of a corresponding unifier σ' is either obvious, or there are hints on the construction. For the SO-cluster rules the following is easy to verify: the (ground) unifier σ' has also an exponent of periodicity not greater than E, since every ground context in the codomain of σ' is already a subcontext of a term or context in the codomain of σ.
If $\sigma(X_i) = Id$, then we select the (CV-elimination) case in the rules. Hence we can assume that $\sigma(X_i) \neq Id$.
The other cases are:

- There is a minimal flat top-SO-cluster K. Then the rule (Solve-flat-cluster) is applied.
  1. Case: $h = 1$. Then all the equations in $EQ(K)$ are of the form $X_1(s_i) \doteq X_1(t_i)$. The transformation replaces these equations by the equations $s_i \doteq t_i$. This is complete, since $\sigma(X_1(s_i)) = \sigma(X_1(t_i))$ implies $\sigma(s_i) = \sigma(t_i)$.
  2. Case: the signature contains only constants and unary function symbols. We show that guessing some X_j as Id is sufficient:
     Let A be the common prefix of $\sigma(X_i)$ for all $X_i \in K$ and let $\sigma(X_i) = AA_i$ for $i = 1, \ldots, h$. Since there are only unary function symbols, there is some j such that $\sigma(X_j) = A$. We select to replace X_j be Id. A unifier σ' of the resulting Γ' is $\sigma'(X_i) := A_i$ for $X_i \in K$, and $\sigma'(X) := \sigma(X)$, $\sigma'(x) := \sigma(x)$, otherwise. This shows completeness in this case.
  3. Case: the signature has at least one function symbol of arity at least two, and $|K| > 1$.
     Let A be the common prefix of all $\sigma(X_i)$, $i = 1, \ldots, h$. If $A = \sigma(X_j)$ for some j, then use selection 1 and replace X_j by Id. A unifier of the output SCUP can be constructed as follows: $\sigma'(X_i) := A_i$ where $\sigma(X_i) = AA_i$ for $i = 1, \ldots, h$ and $\sigma'(X) := \sigma(X)$, $\sigma'(x) := \sigma(x)$, otherwise.
     If A is a proper prefix of $\sigma(X_i)$ for all i, then for $\sigma(X_i) = AA_i$, $i = 1, \ldots, h$, the top function symbol f of A_i must be the same for all A_i, and $k := ar(f) \geq 2$. Let k_i be the first letter of the position of the hole of A_i, $i = 1, \ldots, h$, let $L := \{k_i \mid i = 1, \ldots, h\}$. and let $n := |L|$. It is obvious that $n \leq |K|$.
     Since K is a SO-cluster, for every $i \in \bar{L} := ([1..k] \setminus L)$, and every j: $A_1|_i = A_j|_i$. Let the multi-context $t[\cdot_1, \ldots, \cdot_n]$ be $f(t_1, \ldots, t_k)$, where

$$t_i := \begin{cases} A_1|_i & \text{if } i \in \bar{L} \\ \cdot_{\varphi(i)} & \text{if } i \in L \end{cases}$$

     where φ is a function that arranges the holes in properly ascending sequence. Let F be an n-ary new function symbol. For $i = 1, \ldots, h$, we define $\sigma'(X_i') := A_i|_{k_i}$, $\sigma'(x_{i,j}) := A_i|_{\phi(j)}$, and σ' is like σ, otherwise. This is a unifier of the system after the first replacement in the rule (Solve-flat-cluster). The following steps are obviously complete.
- There is a non-flat, minimal SO-cluster. Then the rule (Solve-non-flat-cluster) is applied to a minimal non-flat top-SO-cluster K. Since the relation $\sim$ in definition 2.7 is an equivalence relation, and there is an equation $X_j(s) \doteq f(t_1, \ldots, t_n)$ in Γ, every $\sigma(X_i)$ for $X_i \in K$ has f as top level function symbol. Thus the imitation instantiation as described in the rule is complete.

□

Lemma 6.8. *The rule (Solve-non-flat-cluster) strictly decreases the measure μ:*

Proof. If selection 1 is used, the number of context variables is strictly decreased. Now consider the case that selection 2 is used.

After decomposing the instantiated equations from $EQ(K)$, the measure is strictly decreased: All the variables $x_{i,j}$ (but no other variables) are removed by the replacement step, hence μ_3 is not increased by the rule. Moreover, the number of context variables is not modified, and the values μ_2 and μ_4 are ∞ before application of the rule, thus cannot be increased.

Let T be the multiset of terms t that occur in the equations of the form $X_i(s) \doteq t$ in $EQ(K)$, where t is not of the form $Y(r)$. Note that T is not empty. Let M be the multiset of the sizes of terms in $term(t)$ corresponding to μ_5, and let M' be the multiset after the application of the rule.

After application of (Decomposition) to the instantiated equations and after the replacement of the variables $x_{i,j}$, we make a comparison of the sizes of M and M'. It is $M \setminus M_1 = M' \cup M_2$ where M_1 are the sizes of the decomposed terms, and M_2 is the multiset of sizes of the added subterms. Since for every $m_2 \in M_2$ there is some $m_1 \in M_1$ with $m_1 > m_2$, the multiset M' is strictly smaller than M in the multiset ordering. Hence the multiset of sizes is strictly decreased. □

Note that the measure μ_3 is required to ensure that the decomposition and variable replacement steps between the rule application do not really increase the measure.

Lemma 6.9. *The rule (Solve-flat-cluster) strictly decreases the measure μ:*

Proof. It is easy to see that either the number of context variables is strictly decreased, or the minimal cardinality of a flat top-SO-cluster is strictly decreased, whereas the components μ_2, μ_3 are not increased. This holds, since in case 2 at least two different indices k_i have to be selected, and the number of different indices corresponds to the number of flat SO-clusters that are generated after the rule as a partition of the old one. □

7 Results

Lemma 7.1. *If the SCUP Γ is not empty, then a transformation rule can be applied.*

Proof. Assume there is no decomposition applicable. If there is an SO-cycle, then an SO-cycle rule is applicable. An action that is always possible is (CV-elimination). Otherwise, by lemma 4.5, there is a top-SO-cluster, and a rule for eliminating SO-clusters can be used, where it is always possible to use the selection (CV-elimination). □

Theorem 7.2. *Stratified context unification is decidable.*

Proof. Given a unifiable SCUP Γ, we fix a minimal unifier and thus also an upper bound E of the exponent of periodicity (see Proposition 2.1).

We have already shown in the previous sections that every rule of the algorithm SCU is sound and complete, and transforms SCUPs into SCUPs. Moreover, since the well-founded measure μ is strictly decreased by every application of a rule, we can apply proposition 3.3. Hence the algorithm SCU decides unifiability of stratified context unification problems. □

Corollary 7.3. *Solvability of one-step rewrite constraints (as defined in [NTT00]) is decidable.*

Remark 7.4. The complexity estimation for stratified context unification that follows from the algorithm SCU is rather bad: There are polynomially many steps that may increase the size usage by an exponential. I.e. an estimated upper bound of space usage may be a tower of exponentials where the tower has polynomial height, and hence the obtained upper bound is non-elementary.

An obvious idea to improve space usage is to use a compact representation of C^e, since it is known that e can be represented in linear space [SSS98]. However, this does not directly imply a polynomial space bound, since (the representation of) C may be too large, and moreover, the algorithm has to be modified considerably.

Some remaining *open questions* are:

- What is the decidability status of context unification?
- What are better upper and lower bounds for the complexity of stratified context unification?
  The currently best known lower bound for the complexity is that it is $\mathcal{NP}$-hard [SSS98]. The author is working on giving a better upper complexity bound for stratified context unification.

8 Acknowledgements

I acknowledge discussions with Klaus Schulz and Joachim Niehren. Thanks to the referees whose hints improved the paper considerably.

References

[BN98] Franz Baader and Tobias Nipkow. *Term Rewriting and All That.* Cambridge University Press, 1998.

[Com93] Hubert Comon. Completion of rewrite systems with membership constraints, part I: Deduction rules and part II: Constraint solving. Technical report, CNRS and LRI, Université de Paris Sud, 1993.

[Com98a] Hubert Comon. Completion of rewrite systems with membership constraints. Part I: Deduction rules. *J. of Symbolic Computation*, 25(4):397–419, 1998.

[Com98b] Hubert Comon. Completion of rewrite systems with membership constraints. Part II: Constraint solving. *J. of Symbolic Computation*, 25(4):421–453, 1998.

[CSTT99] Anne-Cécile Caron, Franck Seynhaeve, Sophie Tison, and Marc Tommasi. Deciding the satisfiability of quantifier free formulae on one-step rewriting. In *Proceedings of the 10th Int. Conf. on Rewriting Techniques and Applications*, volume 1631 of *Lecture Notes in Computer Science*, pages 103–117. Springer-Verlag, 1999.

[Der87] Nachum Dershowitz. Termination of rewriting. *Journal of Symbolic Computation*, 3(1):69–116, 1987.

[DM79] N. Dershowitz and Z. Manna. Proving termination with multiset orderings. *Communications of the ACM*, 22:465–476, 1979.

[Far91] W.A. Farmer. Simple second-order languages for which unification is undecidable. *J. Theoretical Computer Science*, 87:173–214, 1991.

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

[Gut98] C. Gutierrez. Satisfiability of word equations with constants is in exponential space. In *Proceedings FOCS'98*, pages 112–119, Palo Alto, California, 1998. IEEE Computer Society Press.

[Höh97] Claudia Höhl. *Funktionale Implementierung eines Unifikationsalgorithmus für Stratified Second-Order Terme.* Diploma thesis, J.W.Goethe-Universität Frankfurt, 1997. In German.

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

[KP96] Antoni Kościelski and Leszek Pacholski. Complexity of Makanin's algorithms. *Journal of the Association for Computing Machinery*, 43:670–684, 1996.

[Lev96] Jordi Levy. Linear second order unification. In *Proceedings of the 7th International Conference on Rewriting Techniques and Applications*, volume 1103 of *Lecture Notes in Computer Science*, pages 332–346, 1996.

[LV00] Jordi Levy and Margus Veanes. On the undecidability of second-order unification. *Information and Computation*, 159:125–150, 2000.

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

[NPR97a] Joachim Niehren, Manfred Pinkal, and P. Ruhrberg. On equality up-to constraints over finite trees, context unification, and one-step rewriting. In *Proceedings of the International Conference on Automated Deduction*, volume 1249 of *Lecture Notes in Computer Science*, pages 34–48, 1997.

[NPR97b] Joachim Niehren, Manfred Pinkal, and Peter Ruhrberg. A uniform approach to underspecification and parallelism. In *Proceedings of the 35th Annual Meeting of the Association of Computational Linguistics (ACL)*, pages 410–417, Madrid, Spain, 1997.

[NTT00] Joachim Niehren, Sophie Tison, and Ralf Treinen. On rewrite constraints and context unification. *Information Processing Letters*, 74:35–40, 2000.

[Pie73] T. Pietrzykowski. A complete mechanization of second-order type theory. *J. ACM*, 20:333–364, 1973.

[Pla99a] W. Plandowski. Satisfiability of word equations with constants is in NEXPTIME. In T. Leighton, editor, *Proceedings of the Thirty-First Annual ACM Symposium on Theory of Computing (STOC'99)*, pages 721–725, Atlanta, Georgia, 1999. ACM Press.

[Pla99b] W. Plandowski. Satisfiability of word equations with constants is in PSPACE. In *FOCS 99*, pages 495–500, 1999.

[Pre95] Christian Prehofer. *Solving Higher-order Equations: From Logic to Programming.* Ph.D. thesis, Technische Universität München, 1995. In German.

[SG89] Wayne Snyder and Jean Gallier. Higher-order unification revisited: Complete sets of transformations. *J. Symbolic Computation*, 8:101–140, 1989.

[SS94] Manfred Schmidt-Schauß. Unification of stratified second-order terms. Internal Report 12/94, Fachbereich Informatik, J.W. Goethe-Universität Frankfurt, Frankfurt, Germany, 1994.

[SS95] Manfred Schmidt-Schauß. Unification of stratified second-order terms, 1995. available from the author.

[SS98] Manfred Schmidt-Schauß. A decision algorithm for distributive unification. *Theoretical Computer Science*, 208:111–148, 1998.

[SS99] Manfred Schmidt-Schauß. Decidability of bounded second order unification. Technical Report Frank-report-11, FB Informatik, J.W. Goethe-Universität Frankfurt am Main, 1999. submitted for publication.

[SSS98] Manfred Schmidt-Schauß and Klaus U. Schulz. On the exponent of periodicity of minimal solutions of context equations. In *Proceedings of the 9th Int. Conf. on Rewriting Techniques and Applications*, volume 1379 of *Lecture Notes in Computer Science*, pages 61–75, 1998.

[SSS99] Manfred Schmidt-Schauß and Klaus U. Schulz. Solvability of context equations with two context variables is decidable. In *Proceedings of the International Conference on Automated Deduction*, volume 1632 of *Lecture Notes in Computer Science*, pages 67–81, 1999.

[Tre96] Ralf Treinen. The first-order theory of one-step rewriting is undecidable. In *Proceedings of the 7th International Conference on Rewriting Techniques and Applications*, volume 1103 of *Lecture Notes in Computer Science*, pages 276–286. Springer-Verlag, 1996.

[Vor98] Sergei Vorobyov. $\forall\exists^*$-equational theory of context unification is Π_1^0-hard. In *MFCS 1998*, volume 1450 of *Lecture Notes in Computer Science*, pages 597–606. Springer-Verlag, 1998.