Stratified Context Unification is NP-complete*

Jordi Levy[1], Manfred Schmidt-Schauß[2], and Mateu Villaret[3]

[1] IIIA, CSIC, Campus de la UAB, Barcelona, Spain.
http://www.iiia.csic.es/~levy
[2] Institut für Informatik, FB Informatik und Mathematik, Johann Wolfgang Goethe-Universität,
Postfach 11 19 32, D-60054 Frankfurt, Germany.
http://www.ki.informatik.uni-frankfurt.de/persons/schauss/schauss.html
[3] IMA, UdG, Campus de Montilivi, Girona, Spain.
http://ima.udg.es/~villaret

Abstract. Context Unification is the problem to decide for a given set of second-order equations E where all second-order variables are unary, whether there exists a unifier, such that for every second-order variable X, the abstraction $\lambda x.r$ instantiated for X has exactly one occurrence of the bound variable x in r. Stratified Context Unification is a specialization where the nesting of second-order variables in E is restricted.
It is already known that Stratified Context Unification is decidable, NP-hard, and in PSPACE, whereas the decidability and the complexity of Context Unification is unknown. We prove that Stratified Context Unification is in NP by proving that a size-minimal solution can be represented in a singleton tree grammar of polynomial size, and then applying a generalization of Plandowski's polynomial algorithm that compares compacted terms in polynomial time. This also demonstrates the high potential of singleton tree grammars for optimizing programs maintaining large terms.
A corollary of our result is that solvability of rewrite constraints is NP-complete.

1 Introduction

Higher-order logic and higher-order deduction system (see e.g. [Dow01,PS99,Pau94,And86,Hue75]) provide very expressive frameworks and highly developed tools for deduction. One of the operations used in different variants is *higher-order unification* (see [Hue75,Dow01]). A specialization is *second-order unification*, which in turn is a generalization of *first-order unification*, where variables (i.e., second-order variables) at the position of function symbols are permitted in equations. In solving an equation, the second-order variables can stand for an arbitrary first-order term, with holes for plugging in the arguments, which must be terms. In lambda-notation, a second-order

* First and third author's has been partially founded by the CICYT research projects iDEAS (TIN2004-04343) and Mulog (TIN2004-07933-C03-01/03).

variable may be instantiated with a term $\lambda x_1, \ldots, x_n . t$, where t is a first-order term, and the variables x_i also stand for first-order terms. It is known that second-order unification is undecidable, even under severe syntactic restrictions [Gol81,Far91,LV00,LV02].

A variant of second-order unification is *context unification*, which is also a generalization of *string unification*, which is decidable [Mak77] and known to be in PSPACE [Pla04]. Context unification is like second-order unification, where the arity of second-order variables is one, and the possible instantiations of second-order variables are restricted to abstractions where the number of occurrences of the bound variable is one. It is currently open, whether context unification is (un)decidable. A generalization is *linear second-order unification*, see [Lev96], where context variables may have arity more than one, and λ-bindings and bound variables may occur in the terms of the equations. It's decidability is also unknown. A decidable specialization of context unification is *stratified context unification* (SCU) [SS02], which allows only equations, where the nesting of variables obeys a stratification property: For every variable Z, every two positions p_1, p_2 of Z in terms in equations, the sequences of context variables on the paths p_1, p_2 must be the same. It is known that SCU is NP-hard [SSS98] and that the corresponding matching problem is NP-complete [SSS04]. There is a large gap in its precise complexity, since the algorithm for SCU described in [SS02] is non-elementary.

Context unification is also of practical use in computational linguistics [NPR97a,EN00], mainly in the field of compositional semantics of natural language. In fact, SCU subsumes *dominance constraints*, a first-order language that is used to represent scope underspecification [NPR97b,NK01], which has an NP-complete satisfiability problem [KNT98]. Another variant of context unification with interest in computational linguistics is *well-nested context-unification* [LNV05], which restricts the overlap of context variables in the solution; it was recently shown to be in NP.

Another, different, variant of second-order unification with a related algorithmic solution is *bounded second-order unification* (BSOU), with its specialization *monadic second-order unification.* Both problems were recently shown to be NP-complete [LSSV04,LSSV06], using similar methods as in this paper. The difference between BSOU and SCU are semantic: in BSOU the second-order variables may also be instantiated by abstractions without occurrences of the bound variable; and syntactic: the nesting of variables in BSOU may be arbitrary.

In this paper we prove that SCU is in NP, which means that it is NP-complete, closing this complexity gap. The proof-method is interesting in itself: it uses so-called *singleton tree grammars* (STG) [SS05,BLM05,Pla94,LSSV04] as a very general mechanism for compressing terms, i.e. solutions. The known decision algorithm for SCU is adapted and used for showing that the construction of a compressed representation of a size-minimal solution leads to a polynomial-sized STG. Using non-deterministic guessing and an algorithm that can compare compressed terms in polynomial time shows that SCU is in NP. One contribution of compression is to represent C^n with a number n bounded by the exponent of

periodicity in polynomial space. The second contribution, together with the implicit representation of the equation during construction of the STG, is to show that the number of first-order variables remains polynomial. Then "in-NP"- result also implies that the complexity of *rewrite-constraints* is NP-complete (see [NTT00]). The result also demonstrates the high potential of singleton tree grammars for optimizing programs maintaining large terms. The upper complexity bound for SCU also shows the practical potential of SCU, since there is a community that has specialized on providing optimal programs that solve NP-complete problems (see [SAT06]). However, the available upper bound on the order of the involved polynomials is rather high: $O(size(E)^{16})$ for the size of a compressed size-minimal solution, and the upper bound for the time-complexity is further increased by the equality-check.
The paper is structured as follows: After an introduction and the preliminary definitions to explain the basic notions, in Section 3 the compression method using singleton tree grammars (STGs) is described, which permits to represent exponentially large and also exponentially deep terms in polynomial space allowing sharing of terms and contexts. We provide a road-map of the proof in Section 4. In Section 5 we introduce generalized stratified equations. In Sections 6 and 7 it is shown, how the non-elementary SCU decision algorithm can be adapted to the compression method. In Section 8 we summarize the estimations and obtain the result that SCU is in NP.

2 Preliminary Definitions

We consider one base (first-order) type o, and second-order types with the syntax $\tau ::= o \to o \mid o \to \tau$, with the usual convention that $\to$ is associative to the right. We use a *signature* $\Sigma = \bigcup_{i \geq 0} \Sigma_i$, where constants of Σ_i are i-ary, and a set of *variables* $\mathcal{X} = \bigcup_{i=0,1} \mathcal{X}_i$, where variables of $\mathcal{X}_i$ are also i-ary. Variables of $\mathcal{X}_0$ are therefore *first-order variables* and those of $\mathcal{X}_1$ are second-order typed and called *context variables*. We assume that the signature contains at least one 0-ary constant. We denote variables with capital letters Z if it may be first-order as well as context variables, and use the convention that X, Y mean context variables, and x, y, z mean first-order variables. Constants are denoted by lower-case letters a, b, f, g ... respectively. *Second-order terms* are denoted as $s, t, u, v, \ldots$. The set of variables occurring in terms or other syntactic objects is denoted as $FV(\cdot)$. A term without occurrences of free variables is said to be *ground*. The *size* of a term t is denoted $|t|$ and defined as its number of symbols when written in $\beta\eta$-normal form. We use *positions* in terms, denoted p, q, as sequences of non-negative integers following Dewey notation. In $f(t_1, \ldots, t_n)$ or $X(r)$, respectively, the position of the function symbol and the context variable is 0 and the position of the i^{th} argument is i. The symbol at position 0 is also called the *head* of the term. The empty word is notated ϵ, $p \prec q$ denotes the prefix relation, $p \cdot q$ the concatenation, and $t|_p$ the subterm at position p of t.

For ease of notation, we denote linear second-order terms $\lambda x.t$, where x has exactly one occurrence in t as $t[\cdot]$, where $[\cdot]$ indicates the position of the variable, also called *hole*. We call these terms also *contexts*. We denote contexts by upper case letters C, D. If the term s or context D, respectively, is plugged into the hole of $C[\cdot]$, we denote the result as the term $C[s]$ or the context $C[D]$, also denoted as $C \cdot D$. The position of the hole in a context D is called *main path*, denoted $mp(D)$, and the length of the main path is called the *main depth* of D. If $D_1 = D_2[D_3]$ for contexts D_i, then D_2 is called a *prefix* of D_1, and D_3 is called a *suffix* of D_1. Concatenation $C_1[\ldots[C_n]\ldots]$ is written $C_1 \cdot \ldots \cdot C_n$. The notation D^n for a context D and $n \in \mathbb{N}$ means concatenation of n copies of the context D. If $t = D[s]$, then D is a *prefix context* of the term t. A *subcontext* of a context or term is a prefix of some suffix or a prefix context of some subterm. *Second-order substitutions* denoted by greek letters $\sigma, \theta, \ldots$, are functions from terms to terms, defined as usual, where we in addition assume that context variables can only be instantiated with contexts. The application of a substitution σ to a term t is written $\sigma(t)$, where we always assume that the result is beta-reduced.

An instance of the *stratified context unification problem* (SCU) is a *set of equations* $E = \{t_1 \stackrel{?}{=} u_1, \ldots, t_n \stackrel{?}{=} u_n\}$ where t_i and u_i are second-order terms of type o, i.e. terms not containing λ-abstractions. In addition, for every variable $Z \in FV(E)$ and every two positions p_1, p_2 of Z in terms in equations, the sequence of context variables on the path p_1, p_2 is the same. Here we mean that X is *on the path* p in t, iff for some prefix p' of p, $t_{|p'}$ is of the form $X(r)$. The size of an equation E is denoted as $|E|$ and is its number of symbols. We assume that equations are symmetric. A substitution σ is said to be a *unifier* of E, iff for all $i : \sigma(t_i) = \sigma(u_i)$. A unifier σ is said to be a *solution* of E, iff for all i : $\sigma(t_i)$ and $\sigma(u_i)$ are ground. It is easy to see that the following holds:

Lemma 2.1. *Let σ be a solution of the SCU-problem E.*

- *If E contains a function symbol f with $ar(f) \geq 2$, then there is also a solution σ', such that every function symbol g with $ar(g) \geq 1$ occurring in $\sigma'(E)$, also occurs in E.*
- *If for all function symbols f occurring in E we have $ar(f) \leq 1$, if $\sigma(E)$ contains function symbols not in E and h is a function symbol with $ar(h) = 2$, then there is also a solution σ', such that for every function symbol g with $ar(g) \geq 1$ occurring in $\sigma'(E)$: either $g = h$ holds, or g occurs in E.*
- *E is unifiable iff E is solvable.*

It is reasonable to assume that the maximal arity of function symbols is not greater than $size(E)$. In this case the necessary transformations in the proof of Lemma 2.1 can be done in $O(size(E))$. Note that the second case occurs in the equation $X(a) \stackrel{?}{=} Y(b)$, with a solution $\{X \mapsto f(b, [\cdot]), Y \mapsto f([\cdot], a)\}$, but there is no solution using only the symbols occurring in the equation.

It is not a restriction to assume that E contains at least one binary function symbol by adding $f(x, y) = f(x, y)$ to E for a binary function symbol f if necessary. This also allows to restrict E to consist of just one equation.

A solution σ of E is said to be *size-minimal* if it minimizes $\sum_{Z \in FV(E)} |\sigma(Z)|$ among all solutions of E. Size-minimal solutions of a SCU-problem satisfy the exponent of periodicity lemma [Mak77,KP96,SSS98,SS02]:

Lemma 2.2 ([SS02]). *There exists a constant $\alpha \in \mathbb{R}$ such that, for every SCU-problem E, every size-minimal solution σ, every variable X (or x, respectively), contexts u, v and term w, if $\sigma(X) = \lambda y \,.\, u\, v^n(w)$, or if $\sigma(x) = u\, v^n(w)$, and v is not empty, then $n \leq 2^{\alpha|E|}$.*

In the following, we denote by $\mathrm{eop}(\sigma)$ the maximal n such that for nontrivial D, $D^n([\cdot])$ is a subcontext of $\sigma(x)$ or $\sigma(X)(a)$ for variables x, X.

3 Singleton Tree Grammars

We define singleton tree grammars as a generalization of singleton context free grammars (SCFG) [LSSV04,Pla94], extending the expressivity of SCFGs by terms and contexts. This is consistent with [SS05] and [BLM05], and also with the context free tree grammars in [CDG$^+$97], however, it is a special case.

Definition 3.1. *A* singleton tree grammar (STG) *is a 4-tuple $G = (\mathcal{TN}, \mathcal{CN}, \Sigma, R)$, where $\mathcal{TN}$ are tree nonterminals, $\mathcal{CN}$ are context nonterminals, and Σ is a signature of function symbols and constants (the terminals), such that the sets $\mathcal{TN}$, $\mathcal{CN}$, Σ are pairwise disjoint. The set of nonterminals $\mathcal{N}$ is defined as $\mathcal{N} = \mathcal{TN} \cup \mathcal{CN}$. The rules in R may be of the form:*

- *$A ::= f(A_1, \ldots, A_n)$, where $A, A_i \in \mathcal{TN}$, and $f \in \Sigma_n$.*
- *$A_1 ::= C[A_2]$ where $A_1, A_2 \in \mathcal{TN}$, and $C \in \mathcal{CN}$.*
- *$C ::= [\cdot]$.*
- *$C_1 ::= C_2 C_3$, where $C_i \in \mathcal{CN}$.*
- *$C ::= f(A_1, \ldots, A_{i-1}, [\cdot], A_{i+1}, \ldots, A_n)$, where $A_i \in \mathcal{TN}$, $C \in \mathcal{CN}$, $[\cdot]$ is the hole, and $f \in \Sigma$ an n-ary function symbol.*

Let $D' >_G D''$ for two nonterminals D', D'', iff $D' ::= t$, and D'' occurs in t. The STG must be non-recursive, i.e. the transitive closure $>_G^*$ must be terminating. Furthermore, for every non-terminal N there is exactly one rule having N as left hand side. Given a term t with occurrences of nonterminals, the derivation by G is an exhaustive iterated replacement of the nonterminals by the corresponding right hand sides, using the convention for second-order terms. The result is denoted as $w_{G,t}$. In this case we also say, that G defines *$w_{G,t}$. $\hat{E}$ If the grammar G is clear, we omit the index in our notation. As a short hand for $mp(w_C)$ we use $mp(C)$ for context nonterminals C.*

We will also allow variables Z from $\mathcal{X}_0$ and $\mathcal{X}_1$ in the grammar. The convention is that in case there is a rule with left hand side Z, then it is a nonterminal, otherwise we treat Z as terminal.

Definition 3.2. *The* size $|G|$ *of a grammar (STG) G is the number of its rules. The* depth *of a nonterminal D is defined as the maximal number of $>_G$-steps from D. The* depth of a grammar *is the maximum of the depths of all nonterminals, denoted as $depth(G)$.*

As a generalization of the theorem in Plandowski [Pla94,Pla95], in [SS05] and [BLM05], there are proofs of the following theorem, (where we have to simplify away the occurrences of holes):

Theorem 3.3. *Given an STG G, and two tree nonterminals A, B from G, it is decidable in polynomial time depending on $|G|$ whether $w_A = w_B$.*

The following lemmas state how the size and the depth of the grammar are increased by extending the grammar with concatenations, exponentiation, prefixes and suffixes of contexts. The depth/size bounds for these operations are related to balancing conditions for trees. When using log, we mean the binary logarithm. The proofs of the following three lemmas are easy and can be copied from the corresponding proofs for SCFGs in the forthcoming journal version of [LSSV04].

Lemma 3.4. *Let G be an STG defining the contexts $D_1, \dots, D_n$ for $n \geq 1$. Then there exists a STG $G' \supseteq G$ that defines the context $D_1 \cdot \dots \cdot D_n$ and satisfies $|G'| \leq |G| + n - 1$ and $depth(G') \leq depth(G) + \lceil \log n \rceil$.*

Lemma 3.5. *Let G be an STG defining the context D. For any $n \geq 1$, there exists an STG $G' \supseteq G$ that defines the context D^n and satisfies $|G'| \leq |G| + 2 \lfloor \log n \rfloor$ and $depth(G') \leq depth(G) + \lceil \log n \rceil$*

Lemma 3.6. *Let G be an STG defining the context D or term t. For any nontrivial prefix, suffix or subterm D' of the context D, and for every subterm t' of the term t or context D, there exists an STG $G' \supseteq G$ that defines D' or t', respectively, and satisfies $|G'| \leq |G| + depth(G)$ and $depth(G') = depth(G)$.*

Lemma 3.7 covers the case that the main path of the desired prefix context of a term t deviates from the paths as given in the STG. The naïve construction may lead to an exponential blow-up. This case does not occur for words in SCFGs and requires an extra treatment. The prefix context of a context can be constructed as for words, whereas the same construction idea used for constructing a prefix context of a term t may lead to an exponential blow-up for several extensions, since too much rules are required. Hence this case requires an extra treatment.

Lemma 3.7. *Let G be an STG defining the term t. For any nontrivial prefix context D of the term t, there exists an STG $G' \supseteq G$ that defines D and satisfies $|G'| \leq |G| + 2\ depth(G)\ (\log(depth(G)) + 1)$ and $depth(G') \leq depth(G) + 2 + \log(depth(G))$,*

Proof. Let A be the non-terminal symbol defining the term $t = w_A$ and let p be a position in w_A that is the position of the hole of the desired context D. First we show by induction that we can generate a list of context nonterminals that can be concatenated to construct D. The induction is on $depth(A)$.

The base case is that $|p| = 0$ at some depth. In this case the empty context is the result, which is omitted in the list. For the induction step we consider the different possibilities for rules:

1. The rule is $A ::= f(A_1, \ldots, A_n)$ and $p = kp'$. Then we return the context defined by the rule $C_1 ::= f(A_1, \ldots, [\cdot]_k, \ldots, A_n)$, and the list for A_k, p'.
2. The rule is $A ::= C[A_2]$. There are some subcases:
   If p is a prefix of $mp(C)$, then return C_1, constructed such that $p = mp(C_1)$ using Lemma 3.6.
   If p is within A_2, and $p = p_1p_2$, where $p_1 = mp(C)$, then we return C, and the list of contexts generated for A_2, p_2.
   The position p is within C. Then let $p = p_1p_2p_3$, where p_1 is the maximal common prefix of p and $mp(C)$, and $|p_2| = 1$. Then construct C_1 for the prefix of w_C with $p_1 = mp(C_1)$ by Lemma 3.6. Let p_1k with $k \in \mathbb{N}$ be a prefix of $mp(C)$. Let C_3 be a new symbol defining the subcontext of w_C starting at position p_1k using Lemma 3.6. Moreover, there is a defined rule $C_2 ::= f(B_1, \ldots, [\cdot]_k, \ldots B_n)$, corresponding to the subcontext of w_C for position p_1, whose existence can be verified by induction. Since $p_2 \neq k$, we have to define the following new symbols and rules: $A_3 ::= C_3[A_2]$, $C_4 ::= f(B_1, \ldots, [\cdot]_{p_2}, \ldots, B_{k-1}, A_3, B_{k+1}, \ldots, B_n)$. Then return C_1, C_4 and the list generated for B_{p_2}, p_3.

Summarizing, we obtain a list of contexts of length at most $2\mathit{depth}(G)$, which can be concatenated defining a new symbol C_D. An upper bound on the total number of new rules is $(2\log(\mathit{depth}(G)) + 2) * \mathit{depth}(G)$, since the induction hypothesis in case 2 is called for $\mathit{depth}(A) - 2$. Notice that the depth of all the contexts that we build up is bounded by $\mathit{depth}(G)+1$ because of the construction of C_4, hence the depth of C_D is at most $\mathit{depth}(G) + 2 + \log(\mathit{depth}(G))$, which is the depth contribution of the final concatenation. □

3.1 Estimations for Several Grammar Extensions

Definition 3.8. *Let G, G' be STGs, let $M \in \mathbb{R}$ with $M \geq 2$. Then we write $G \to_{sd(M)} G'$ for a* grammar extension *by size and depth, iff*

$$\begin{aligned} |G'| &\leq |G| + 3\log(\mathit{depth}(G))\mathit{depth}(G) + 2M \\ \mathit{depth}(G') &\leq \mathit{depth}(G) + \log(\mathit{depth}(G)) + M \end{aligned}$$

As an abbreviation, we write $G \to^k_{sd(M)} G'$, iff $G \to_{sd(M)} G_1 \ldots G_{k-1} \to_{sd(M)} G'$ for appropriate STGs G_i and an integer $k \geq 2$.

Proposition 3.9. *Let G, G' be STGs, let $M \in \mathbb{R}$, such that $G \to^n_{sd(M)} G'$. Then with $M' = \max(M, \mathit{depth}(G))$ and $\beta(M, n) := (n + 2)M' + n\log(M') + n^2$:*

$$\begin{aligned} |G'| &\leq |G| + 3n\beta(M', n)\log(\beta(M', n)) + 2Mn \\ \mathit{depth}(G') &\leq \beta(M', n) \end{aligned}$$

Proof. Let $G = G_0, G_1, \ldots G_n = G'$ be a sequence of STGs, such that for every $i = 0, \ldots, n-1$: $G_i \to_{sd} G_{i+1}$. To verify the bound for $\mathit{depth}(G_n)$, let $d_i := \mathit{depth}(G_i), i = 1, \ldots, n$. Then $d_{i+1} = d_i + \log(d_i) + M$, which implies $\mathit{depth}(G_n) \leq d_0 + nM + \sum(\log(d_i))$. Using $\log(d_i + a) \leq \log(d_i) + a/d_i$, it follows that $\log(d_{i+1}) -$

$\log(d_i) \leq 1$ for $i \geq 2$. Then we obtain $depth(G_n) \leq d_0 + nM + n(2 + \log(M')) + n^2 \leq (n+2)M' + n\log(M') + n^2$. The bound for $|G_n|$ can be derived from $|G_n| \leq |G_0| + 3\sum_i \big(\log(\beta(M',n)) * \beta(M',n)\big) + 2Mn$. $\square$

Corollary 3.10. *Let G be an STG, and G' be constructed from G by n grammar extensions according to Lemmas 3.4, 3.5, 3.6 and 3.7. Assume $M = max(\lceil\log(\text{eop})\rceil, k)$, where k is the maximal number of concatenated nonterminals in Lemma 3.4, and the exponent in 3.5 is bounded by* eop. *For an initial system of equations E_0, let $M = O(|E_0|)$, $|G| = O(|E_0|)$, $depth(G) = O(|E_0|)$, and $n = O(|E_0|^h)$, where $h > 1$. Then*

4 Overview of the Proof Idea

Given a solvable stratified equation, we show that we can construct a polynomial-sized solution and test it also in polynomial time. The second part, i.e. the test, is delegated to STGs. Showing the first part is the new contribution: Given a solvable stratified equation, the idea is to first fix a size-minimal solution σ, and then to compute a compressed representation. The algorithm in [SS02] is used, however, using a representation, where nonterminals from an STG and variables are also allowed in equations as abbreviations for larger terms. The solution σ will be used as a guide to perform a step-by-step computation of a representation of the solution σ together with an STG. We do not care about the efficiency of this algorithm, since only the size of the computed representation is of interest.

The computation will proceed as follows: After some initialization, the state consists of three components: an equation E, an STG G, and the solution σ. Variables from E may be terminals or non-terminals in the STG. As in [SS02], there will be a distinction between the cases:

1. there is no chain of equations that constitutes a cycle, or
2. there is at least one cycle of equations.

If $FV(E)$ is empty, the construction is finished. In the first case, we extend the partial solution by using a decomposition-like detection of decomposable subequations. In the second case, we use the algorithm in [SS02] and show that a cycle allows to compute a complete instantiation of at least one context variable. The algorithm will terminate and constructs an at most polynomial size representation of σ by an STG.

5 Generalized Stratified Equations

In the following we compact partial solutions as well as equations using STGs, where STG-symbols are permitted in equations. We also allow rules of the form $C ::= C'$, which does not extend the expressive power, since it can be easily removed later by the appropriate replacements in the STG.

5.1 Basic Definitions

Definition 5.1. *Let G be an STG, and E be a single stratified equation, where symbols from G are permitted in E. Then (E, G) is a* generalized stratified equation (GSE). *We denote the set of variables occurring in E after expansion of nonterminals using G by $FV_G(E)$, where the variables that are nonterminals in G are not considered.*

We fix a size-minimal solution σ_0 for the initial stratified equation $E_{\text{initial}} = \{u_1 \stackrel{?}{=} u_2\}$, and denote its size by M_0, and the exponent of periodicity bound of σ_0 according to Lemma 2.2 by eop. Let $W := FV(E_{\text{initial}})$. The partial solution, denoted by θ, is always given by the right hand sides in G of the variables in W, i.e. by $\theta(Z) := w_Z$ for all $Z \in W$. The corresponding initial GSE is (E_0, G_0), where G_0 encodes the terms u_1, u_2 as tree nonterminals U_1, U_2, and $E_0 := \{U_1 \stackrel{?}{=} U_2\}$. The initial state of the construction is $((E_0, G_0), \sigma_0)$. A solution σ of an intermediate GSE (E, G) is called *correct*, iff $\sigma\theta(Z) = \sigma_0(Z)$ for all variables $Z \in W$. The construction of the solution uses a state $((E, G), \sigma)$ where σ is a correct solution. For all correct solutions σ, we will have $\text{eop}(\sigma) \leq \text{eop}$, since all concerned subcontexts are also subcontexts of $\sigma_0(Z)$ for $Z \in W$.

A *surface position* in a term t is a position p in t, such that for all prefixes p' of p: $t_{|p'}$ has a function symbol as head. We denote w_{U_i} as w_i for $i = 1, 2$ in the following. We repeat the definitions in [SS02] adapted to our representation.

Definition 5.2 (cycles). *Let (E, G) be a GSE. Let the* surface equations *of (E, G) be all equations $w_{1|p} \stackrel{?}{=} w_{2|p}$, that can be derived by decomposition from $w_1 \stackrel{?}{=} w_2$, where p is a surface position of w_1 and w_2. We denote the surface equations by surfE(E). Let $\approx$ be an equivalence relation on $FV_G(E)$ generated by all the relations $x \approx Y$ for surface equations $x \stackrel{?}{=} Y(s)$, $x \approx y$ for surface equations $x \stackrel{?}{=} y$, and $X \approx Y$ for surface equations $X(s) \stackrel{?}{=} Y(t)$. Let $\succ$ on $FV_G(E)$ be defined as follows: $x \succ Z$ if $x \stackrel{?}{=} s$ is in surfE(E), the head of w_s is a function symbol, and Z occurs in w_s at a surface position, i.e. there is some surface position p such that $w_{s|p} = Z(r)$ for some r; $X \succ Z$ if $X(r) \stackrel{?}{=} s$ is in surfE(E) and the head of w_s is a function symbol. Let $\succeq$ be the smallest pre-order generated by $\approx$ and $\succ$. If for all $Z_1, Z_2 \in FV_G(E)$: $Z_1 \succ Z_2$ implies that $Z_1 \not\preceq Z_2$, then (E, G) is* cycle-free*, otherwise, (E, G) is* cyclic*.*
A cycle *is a sequence of surface equations, which in expanded form is as follows: $Z_1(\ldots) \stackrel{?}{=} D_1(Z_2(\ldots)), Z_2(\ldots) \stackrel{?}{=} D_2(Z_3(\ldots)), \ldots, Z_h(\ldots) \stackrel{?}{=} D_h(Z_1(\ldots))$, where Z_i may be context-variables or first-order variables, D_i is a context for all i, and at least one D_i is a nontrivial context. The indicated occurrences directly correspond to the definition of $\approx$ and $\succ$. However, note that there may be different occurrences of the Z_i on the right hand side, and that D_i may not be unique. The* length *of the cycle is the number of the equations occurring in it.*

Lemma 5.3 (Occurs-check). *In a solvable GSE (E, G), there is no cycle where all variables $Z_1, \ldots, Z_h$ are first-order variables.*

We define an ordering that will be reduced by mimicking the SCU-algorithm from [SS02] and use it for the construction of a representation of σ_0.

Definition 5.4. *Given a GSE (E, G), the measure $\mu(E)$ is the lexicographic combination $\langle \mu_1(E), \mu_2(E), \mu_3(E) \rangle$ of the following components:*

1. $\mu_1(E) = |FV_G(E)|$.
2. $\mu_2(E) = 0$ *if (E, G) is cyclic, and 1, otherwise.*
3. $\mu_3(E) = \mu_1(E) - |\{[Z]_\approx \mid Z \in FV_G(E)\}|$ *if E is not cyclic, and* 0 *otherwise.*
4. $\mu_4(E)$ *is the number of variables in $FV_G(E)$ that are not $\succ$-maximal.*

The transformations will never increase $|FV_G(E)|$ and they will transform a stratified equation into a stratified equation (see [SS02]).

6 GSE without Cycles

The SCU-algorithm in [SS02] treats systems of equations without cycles by iteratedly guessing and instantiating parts of the solution σ. The potential number of these guessing, instantiating and decomposition steps in that paper may be exponential. We have to avoid steps that lead to unnecessary constructions of symbols and rules in G. Hence we have to adapt the algorithm given in [SS02] to our compressing method. Let a *var-term* be a term of the form x or $X(r)$.

Algorithm 6.1 (Rule: Transform-non-cyclic GSE). Let (E, G) be a non-cyclic GSE with $E = \{U_1 \stackrel{?}{=} U_2\}$, $w_i = w_{U_i}$, for $i = 1, 2$, and let σ be a correct solution. Depending on E, there are several possibilities:

1. w_1, w_2 are ground and $w_1 = w_2$. Then stop further instantiation.
2. There is a context variable $X \in FV_G(E)$ with $\sigma(X) = [\cdot]$. Then add $X ::= [\cdot]$ to G.
3. Assume cases (1) and (2) are not applicable.
   Let p be a surface position in w_1 and w_2 such that p is a position of a first-order variable in w_1 or w_2 which is also in $FV_G(E)$. Also assume that $w_{1|p} \neq w_{2|p}$. W.l.o.g. let $w_{1|p}$ be the first order variable, say x. Add a nonterminal A defining $w_{2|p}$, and add $x ::= A$ to the grammar.
4. Assume that the cases (1) – (3) are not applicable. Then let V be a $\succeq$-maximal $\approx$-equivalence class in $FV_G(E)$, that is in addition not $\succeq$-minimal. Note that V consists only of context variables. Let s be a term with a function symbol as head (such an s must exist), such that there is a surface position p and $w_{1|p} = s$, and $w_{2|p} = X(r)$ for some $X \in V$.
   Let q be the maximal position such that q is a prefix of all main paths of $\sigma(X)$ for all $X \in V$, and such that q is a surface position in s. There are some subcases:
   (a) q is the main path of some context $\sigma(X)$ where $X \in V$. Then construct A_s with $w_{A_s} = s$, and the symbol C for the prefix of A_s with main path q. For all $X \in V$, add $X ::= CX'$ or $X ::= C$ to G, where the X' are new context variables. The latter case is used iff $\sigma(X)$ has main path q.
   (b) If $s_{|q}$ is a var-term, then construct A_s with $w_{A_s} = s$, and the prefix symbol C of A_s with main path q. For all $X \in V$, add $X ::= CX'$ to G.

   (c) Case (4a) does not apply and $s_{|q}$ is not a var-term. This is the situation where the contexts go into different directions. Then let $V = \{X_1, \ldots, X_n\}$ and for all $i = 1, \ldots, n$ let q_i be a position of length 1, such that qq_i is a prefix of the main path of $\sigma(X_i)$. Construct A_s with $w_{A_s} = s$, and for every $i = 1, \ldots, n$ the prefix context C_i of A_s with main path qq_i. For all $X_i \in V$, add $X_i ::= C_i X'_i$ to G where X'_i are new context variables.

5. Assume that the cases (1) – (4) are not applicable. Let $V = \{X_1, \ldots, X_n\}$ be a $\succeq$-maximal $\approx$ −equivalence class in $FV_G(E)$, that is in addition $\succeq$-minimal. Note that V consists only of context variables. For $i = 1, \ldots, n$ let q_i be a position of length 1 that is a prefix of $\sigma(X_i)$. Minimality of σ_0 implies that $|\{q_i \mid i = 1, \ldots, n\}| \geq 2$. Since $\sigma(X_i) \neq [\cdot]$, there is a function symbol f, also occurring in E, which is the head of all $\sigma(X_i)$. Construct the context symbols C_i with rules $C_i ::= f(A_{i,1}, \ldots, [\cdot]_{q_i}, \ldots, A_{i,n})$, where q_i is the first integer on the main path of $\sigma(X_i)$. The symbol $A_{i,j}$ stands for a constant a_j, if for all i: $\sigma(X_i)_{|j} = a_j \in \Sigma_0$. Let $J \subseteq \{1, \ldots, n\}$ be the indices, for which this is false. Note that $|J| \geq 2$. For indices $j \in J$, let $A_{i,j}$ be new first-order variables. Define the rules $X_i ::= C_i(X'_i)$ for new context variables X'_i.
   Since we have added first-order variables, we apply now the decomposition in (3) for all positions of $A_{i,j}$ for $j \in J$, successively, until for every first-order variable $A_{i,j}$, there is a rule in G. These rules are only of the form $A_{i,j} ::= A_{i',j}$ or $A_{i,j} ::= X'_{i'}(r)$.

In every case, we define the new solution $\sigma'(Z) := \sigma(r)$ for variables Z, if $Z ::= r$ is the new rule for Z, perhaps in several steps.

Case (5) is the key difference between SCU and context unification: in SCU the context variables X_i are only at the surface, and so an instantiation can be guessed, whereas in CU the occurrences of X_i may also be elsewhere, and guessing and instantiating these variables in general makes no progress in solving the equation.

Theorem 6.2. *If $((E, G), \sigma)$ is a GSE with a correct solution σ, then the transformations in (Transform-non-cyclic GSE) are correct. The order μ is strictly decreased. The number of grammar extensions of a single execution can be estimated for the different cases as follows:*
(2) *requires 1 extension step,* (3) *requires 2 extension steps,* (4) *requires at most* $2 + 2M_0$ *extension steps, and* (5) *requires at most* $2M_0 + M_0^2$ *grammar extension steps.*[1]

Proof. The standard cases for correctness follow from [SS02]. The only non-standard case is in case (5). If there is an equivalence class $V = \{X_1, \ldots, X_n\}$ that is $\succeq$-maximal and $\succeq$-minimal, consisting only of context variables, and $\sigma(X_i)$ is not trivial for all i, then the common prefix of the holes of all $\sigma(X_i)$ must have length 0, since σ_0 was chosen as size-minimal. Hence the algorithm

[1] $M_0 := |E_0|$ is defined in subsection 5.1

part (5) covers all cases. Moreover, the class V will be replaced by at least two $\approx$-classes after the execution. Since also the number of variables is not increased, in this case, $\mu(\cdot)$ is strictly decreased. $\mu(\cdot)$ is also strictly decreased in all other cases: Either a cycle is introduced, or the number of variables is strictly decreased in cases (2), (3), and (4a). In case (4b), the number of non-maximal variables is strictly decreased, and in case (4c), the number of equivalence classes is increased, hence μ_3 is strictly decreased.

The number of grammar extensions as given in the theorem can be checked by simply scanning the cases of the algorithm. $\square$

7 GSE with Cycles

Given a cyclic GSE (E, G) and a solution σ, we mimic the algorithm in [SS02]. A cycle K in the expanded representation is of the form

$$X_1(s_1) \stackrel{?}{=} D_1(X_2(t_1)), \ldots, X_{h-1}(s_{h-1}) \stackrel{?}{=} D_{h-1}(X_h(t_{h-1})), X_h(s_h) \stackrel{?}{=} D_h(X_1(t_h))$$

provided there are no first-order variables in the cycle. Note that the contexts D_i are not necessarily unique, since there may be further occurrences of X_{i+1}, respectively X_1 in D_h.
We use the measure $\chi(K) = (\chi_1(K), \chi_2(K))$ for a cycle K as above, where $\chi_1(K)$ is the length of the cycle K, and $\chi_2(K)$ is h minus the maximal number i, such that $D_1, \ldots, D_i$ are trivial contexts, perhaps after a rotation of the cycle.

Algorithm 7.1 (Rule: Solve-cycle). This step is applied to a cycle K that is minimal w.r.t. χ. There are four cases.

1. There is a first-order variable in the cycle. Then apply step (3) of rule 6.1.
2. There is a context-variable X with $\sigma(X) = [\cdot]$. Then apply step (2) of rule 6.1 to eliminate one context variable.
3. Some D_i for $i \neq h$ is nontrivial and (1) and (2) do not apply. Then let k be the minimal index such that D_k is nontrivial. Let q be the maximal common prefix of $mp(D_k)$ and of $mp(\sigma(X_i))$ for $i = 1, \ldots, k$. Construct C that represents the prefix context of D_k with hole at position q, and add $X_i ::= CX'_i$ for $i = 1, \ldots, k$.
4. The cases (1) – (2) do not apply, and only D_h is nontrivial. Then let q be the maximal common prefix of $mp(D_k^{\mathrm{eop}+1})$ and $mp(\sigma(X_i))$ for $i = 1, \ldots, h$. Construct C_0 as the subcontext of $D_k^{\mathrm{eop}+1}$ with hole at position q.
   (a) The position q is the main path of some $\sigma(X_i)$ where $i = 1, \ldots, h$. For all $i = 1, \ldots, h$ add $X_i ::= C_0 X'_i$ or $X_i ::= C_0$ to G; the latter if $\sigma(X_i) = C_0$.
   (b) Otherwise, for $i = 1, \ldots, h$ let qq_i be the prefix of $mp(\sigma(X_i))$ with $|q_i| = 1$. Note that all contexts $\sigma(X_i)_{|q}$ have the same function symbol f as head, which also occurs in E. Construct the contexts symbols C_i with rules $C_i ::= f(x_{i,1}, \ldots, [\cdot]_{q_i}, \ldots, x_{i,n})$, and $C'_i ::= C_0 C_i$, where $x_{i,j}$ are fresh first-order variables. Define the rules $X_i ::= C'_i(X'_i)$ for new context variables X'_i. Then apply the step (3) of rule 6.1 several times until all the variables $x_{i,j}$ are instantiated.

Theorem 7.2. *Given a GSE (E, G) with cycles and a solution σ with $\mathrm{eop}(\sigma) \leq \mathrm{eop}$. Then it is possible to construct a GSE (E', G'), such that $\mu_1(E', G') < \mu_1(E, G)$, and there are at most $O(M_0^4)$ grammar extension steps necessary until this happens, and there is a correct solution σ' with $\mathrm{eop}(\sigma') \leq \mathrm{eop}$.*

Proof. Since we have mimicked the algorithm in [SS02], we have only to check the number of grammar-extension steps. In every step, a cycle can be chosen that is χ-minimal. We estimate the number of applications until $\mu_1(\cdot)$ is strictly decreased. First, the number of applications of (Solve-cycle) is at most M_0^2, since in case $\mu_1(\cdot)$ is unchanged, χ of a minimal cycle is strictly decreased, which follows from [SS02]. The number of grammar-extension of one application can be estimated as follows: The construction of C_i requires $O(M_0)$ grammar extensions, the removal of the variables in case (4b) requires $O(M_0^2)$ extensions. Since we have at most M_0^2 executions and every execution requires $O(M_0^2)$ grammar extensions, we have $O(M_0^4)$ as an upper bound. □

8 Upper Bound on the Complexity of Stratified Context Unification

Lemma 8.1. *There are at most $O(M_0^3)$ steps that strictly decrease the order μ.*

Proof. Three components in the lexicographic ordering are at most M_0, and the remaining one is bound by a constant.

Summarizing the estimations results in a NP upper bound:

Theorem 8.2. *Stratified context unification is in NP.*

Proof. Given a solvable stratified context unification problem E_{initial} of size M_0 and a size-minimal solution σ_0, we know that there is an upper bound on the exponent of periodicity, denoted as eop, which is of order $O(M_0)$. Theorem 7.2 shows that there are at most $O(M_0 * M_0^4)$ grammar extensions due to cyclic GSE. Theorem 6.2 and Lemma 8.1 show that there are at most $O(M_0^3 * M_0^2)$ grammar extensions due to non-cyclic GSE.

This means the number of grammar extensions is of order $O(M_0^5)$. Since the initial grammar has size and depth of order $O(M_0)$, Corollary 3.10 shows that the size of the final STG is of order $O(M_0^{3*5+1}) = O(M_0^{16})$.

Now the complexity estimation is as follows: Given E_{initial}, we compute E_0, G_0 as above. Then we guess a solution θ represented by an STG that is an extension of G_0 of size at most $O(M_0^{16})$. The variables can be used exactly as we had done it in the computation. The final equation is of the form $U_1 \stackrel{?}{=} U_2$ where U_1, U_2 are defined by the STG. Rules of the form $C ::= [\cdot]$, and $C ::= C'$, which may be generated by the guessing can easily be removed by performing the appropriate replacements in the STG. Now Theorem 3.3 (see [SS05,BLM05]) shows that we can decide equality of U_1, U_2 in polynomial time. This shows that stratified context unification is in NP. □

Corollary 8.3. *Stratified context unification is NP-complete and solvability of rewrite constraints (as defined in [NTT00]) is NP-complete.*

Proof. The first claim follows from Theorem 8.2 and from NP-hardness shown in [SSS98]. The second claim follows from the equivalence proof in [NTT00].

It is not clear whether unifiability of generalized stratified context-unification problems (E, G) is in NP, since the usual encoding does not produce a stratified unification problem. However, the following is easy:

Corollary 8.4. *Unifiability of generalized SCU-problems is in NEXPTIME.*

Proof. We can first expand the equation to get rid of the STG, which results in an at most exponentially large SCU-problem, and then we apply Theorem 8.2.

9 Conclusion and Further Research

We have shown that stratified context unification is NP-complete by exploiting compaction of terms and polynomial comparison of the compactions using singleton tree grammars. This also determines the complexity of rewrite constraints to be NP-complete. The result in this paper is a hint that the complexity of distributive unification, which was shown to be decidable in [SS98], may also be in NP, since the algorithm can be seen as an extension of the algorithm for SCU. The compressing mechanism via STGs deserves further investigations to obtain better bounds for the operations on STGs.

References

[And86] P. Andrews. *An introduction to mathematical logic and type theory: to truth through proof.* Academic Press, 1986.

[BLM05] G. Busatto, M. Lohrey, and S. Maneth. Efficient memory representation of XML documents. In *Proc. DBPL 2005*, LNCS 3774, pages 199–216, 2005.

[CDG$^+$97] H. Comon, M. Dauchet, R. Gilleron, F. Jacquemard, D. Lugiez, S. Tison, and M. Tommasi. Tree automata techniques and applications. Available on: `http://www.grappa.univ-lille3.fr/tata`, 1997. release 1.10.2002.

[Dow01] G. Dowek. Higher-order unification and matching. In A. Robinson and A. Voronkov, editors, *Handbook of Automated Reasoning*, volume II, chapter 16, pages 1009–1062. Elsevier Science, 2001.

[EN00] K. Erk and J. Niehren. Parallelism constraints. In *RTA-11*, LNCS 1833, pages 110–126, 2000.

[Far91] W. M. Farmer. Simple second-order languages for wich unification is undecidable. *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.

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

[KNT98] A. Koller, J. Niehren, and R. Treinen. Dominance constraints: Algorithms and complexity. In *3rd LACL '98*, LNCS 2014, pages 106–125, 1998.

[KP96] A. Kościelski and L. Pacholski. Complexity of Makanin's algorithm. *Journal of the ACM*, 43(4):670–684, 1996.

[Lev96] J. Levy. Linear second order unification. In *RTA-7*, LNCS 1103, pages 332–346, 1996.

[LNV05] J. Levy, J. Niehren, and M. Villaret. Well-nested context unification. In *CADE 2005*, LNCS 3632, pages 149–163, 2005.

[LSSV04] J. Levy, M. Schmidt-Schauß, and M. Villaret. Monadic second-order unification is NP-complete. In *RTA-15*, LNCS 3091, pages 55–69, 2004.

[LSSV06] J. Levy, M. Schmidt-Schauß, and M. Villaret. Bounded second-order unification is NP-complete. to appear in Proc. RTA' 06, 2006.

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

[LV02] J. Levy and M. Villaret. Currying second-order unification problems. In *RTA-13*, LNCS 2378, pages 326–339, 2002.

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

[NK01] J. Niehren and A. Koller. Dominance constraints in context unification. In *LACL'98*, LNAI 2014, pages 199–218, 2001.

[NPR97a] J. Niehren, M. Pinkal, and P. Ruhrberg. On equality up-to constraints over finite trees, context unification, and one-step rewriting. In *CADE-14*, LNCS 1249, pages 34–48, 1997.

[NPR97b] J. Niehren, M. Pinkal, and P. Ruhrberg. A uniform approach to underspecification and parallelism. In *35th ACL'97*, pages 410–417, Madrid, 1997.

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

[Pau94] Lawrence C. Paulson. *Isabelle*. LNCS 828. Springer-Verlag, 1994.

[Pla94] W. Plandowski. Testing equivalence of morphisms in context-free languages. In J. van Leeuwen, editor, *2nd ESA'94*, LNCS 855, pages 460–470, 1994.

[Pla95] W. Plandowski. *The Complexity of the Morphism Equivalence Problem for Context-Free Languages*. PhD thesis, Dept. of Mathematics, Informatics and Mechanics, Warsaw University, 1995.

[Pla04] W. Plandowski. Satisfiability of word equations with constants is in PSPACE. *Journal of the ACM*, 51(3):483–496, 2004.

[PS99] F. Pfenning and C. Schürmann. System description: Twelf - a meta-logical framework for deductive systems. In H. Ganzinger, editor, *Proc. CADE-16*, LNAI 1632, pages 202–206. Springer-Verlag, 1999.

[SAT06] 2006. `http://www.satlive.org/`.

[SS98] M. Schmidt-Schauß. A decision algorithm for distributive unification. *TCS*, 208:111–148, 1998.

[SS02] M. Schmidt-Schauß. A decision algorithm for stratified context unification. *Journal of Logic and Computation*, 12(6):929–953, 2002.

[SS05] M. Schmidt-Schauß. Polynomial equality testing for terms with shared substructures. Frank report 21, Institut für Informatik. FB Informatik und Mathematik. J. W. Goethe-Universität Frankfurt am Main, November 2005.

[SSS98] M. Schmidt-Schauß and K. U. Schulz. On the exponent of periodicity of minimal solutions of context equations. In *9th RTA'98*, LNCS 1379, pages 61–75, Tsukuba, Japan, 1998.

[SSS04] M. Schmidt-Schauß and J. Stuber. On the complexity of linear and stratified context matching problems. *Theory of Computing Systems*, 37:717–740, 2004.