One-context Unification with STG-Compressed Terms is in NP*

Carles Creus[1], Adrià Gascón[1], and Guillem Godoy[1]

1 **Universitat Politècnica de Catalunya, departament de Llenguatges i Sistemes Informàtics, Jordi Girona 1, Barcelona, Spain**
ccreuslopez@gmail.com adriagascon@gmail.com ggodoy@lsi.upc.edu

Abstract

One-context unification is an extension of first-order term unification in which a variable of arity one standing for a context may occur in the input terms. This problem arises in areas like program analysis, term rewriting and XML processing and is known to be solvable in nondeterministic polynomial time. We prove that this problem can be solved in nondeterministic polynomial time also when the input is compressed using Singleton Tree Grammars (STG's). STG's are a grammar-based compression method for terms that generalizes the directed acyclic graph representation. They have been recently considered as an efficient in-memory representation for large terms, since several operations on terms can be performed efficiently on their STG representation without a prior decompression.

1998 ACM Subject Classification F.2.2 Nonnumerical Algorithms and Problems

Keywords and phrases Term Unification; Compression; Grammars

Digital Object Identifier 10.4230/LIPIcs.RTA.2012.149

Category Regular Research Paper

1 Introduction

Term unification is a basic operation in many areas of computer science, especially in those related to logic (see [1] for a survey). Unifying two terms s, t corresponds to find a substitution σ for the variables occurring in the equation $s \doteq t$ such that $\sigma(s) = \sigma(t)$ holds. The particular case when one of the terms has no occurrences of variables is called matching.

A simple case of term unification is called first-order unification. In this case variables occur at leaf positions and stand for terms. Both the first-order unification and matching problems have applications in the context of automated deduction, functional and logic programming, rewriting, and pattern matching. Moreover, several variants of first-order unification have been studied to tackle problems arising in those areas, see [1]. A remarkable extension is unification modulo theories. In this notion of unification, equality between terms is interpreted under equational theories such as associativity, commutativity, and distributivity, among others.

Another widely considered notion of unification allows variables of arity one standing for contexts, in addition to variables at leaf positions. This extension is called context unification,

* The authors were supported by Spanish Ministry of Education and Science by the FORMALISM project (TIN2007-66523). The first author was also supported by an FPI-UPC grant. The second author was also supported by an FPU grant from the Spanish Ministry of Education. The last author was also supported by the Spanish Ministry of Science and Innovation SweetLogics project (TIN2010-21062-C02-01).

23rd International Conference on Rewriting Techniques and Applications (RTA'12).
Editor: A. Tiwari; pp. 149–164

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

which is a particular case of second-order unification. While second-order unification is known to be undecidable [8], decidability of context unification is still open. However, some interesting results have been found for more particular cases with application in computational linguistics [10, 4, 12]. One-context unification is the particular case when only one context variable, possibly with many occurrences, may appear in the input terms. This problem can be solved in nondeterministic polynomial time [6] but it is not known whether it is NP-hard. One of the motivations for this specific variant is its close relation to interprocedural program analysis [9]. The goal of this kind of analysis is to compute all simple invariants of imperative procedural programs. Other interesting applications of one-context unification and matching arise in searching/extracting information from tree data structures. For example, a simple matching equation of the form $F(s) \doteq t$, where F is the context variable, t is ground, and s may contain first-order variables but it does not contain occurrences of F, corresponds to search instances of s within t. In the context of XML processing, combinations of unificaton modulo commutativity and context unification are used for querying XML documents with a logic programming approach [2].

Other relevant variants of unification are related to term compression. Some applications dealing with trees, especially in the context of XML processing, require some kind of succint representation for terms. For this reason, in addition to the well-known Directed Acyclic Graph representation (DAG), several compressed in-memory representations have been developed [18, 15]. Grammar-based compression techniques were initially applied to words [17] and led to important results in string processing. Besides its direct applications in data compression, this kind of representations has also been useful as a technique for complexity analysis [11]. This compression mechanisms were extended from words to terms in [3] to introduce Singleton Tree Grammars (STG's), which allow to represent terms with exponential size and height in linear space. STG's are useful in the context of XML tree structure compression [15], tree automata [16], XPATH [14], and unification theory [12]. Moreover, polynomial time algorithms for first-order unification and matching for the case when the input is compressed using STG's have been developed and implemented [5, 7].

In this work we extend the result in [6] proving that one-context unification can be solved in nondeterministic polynomial time even in an STG-compressed setting.

2 Preliminaries

2.1 Terms, Contexts and Substitutions

A *ranked alphabet* $\mathcal{F}$ is a (finite) set of function symbols with arity. Symbols in $\mathcal{F}$ of arity 0, called *constants*, are denoted by a, b; nonconstant symbols are denoted by g, h. A *set of variables* $\mathcal{V}$ is a ranked alphabet containing symbols of arity 0, called *first-order variables* and denoted by x, y, and symbols of arity 1, called *context variables* and denoted by F. We assume that $\mathcal{F}$ and $\mathcal{V}$ are always disjoint. We use α to refer to an element of $\mathcal{F} \cup \mathcal{V}$. The set $\mathcal{T}(\mathcal{F}, \mathcal{V})$ of *terms* over $\mathcal{F}$ and $\mathcal{V}$ is the smallest set such that $\alpha(t_1, \ldots, t_m)$ is in $\mathcal{T}(\mathcal{F}, \mathcal{V})$ whenever $\alpha \in \mathcal{F} \cup \mathcal{V}$, the arity of α is m, and $t_1, \ldots, t_m \in \mathcal{T}(\mathcal{F}, \mathcal{V})$. By s, t, u we denote terms in $\mathcal{T}(\mathcal{F}, \mathcal{V})$. The set of variables occurring in a term t is denoted by $\mathsf{Vars}(t)$.

A *position* is a sequence of natural numbers. The symbol λ denotes the empty sequence, also called the *root position*, and $p.p'$ denotes the concatenation of the positions p and p'. The set of positions of a term t, denoted by $\mathsf{Pos}(t)$, is defined recursively as $\mathsf{Pos}(\alpha(t_1, \ldots, t_m)) = \{\lambda\} \cup \{i.p \mid i \in \{1, \ldots, m\} \wedge p \in \mathsf{Pos}(t_i)\}$. The *length* of a position is denoted by $|p|$. Note that $|\lambda| = 0$ and $|i.p| = 1 + |p|$ hold. A position p_1 is a *prefix* of a position p, denoted $p_1 \leq p$, if there is a position p_2 such that $p_1.p_2 = p$ holds. Also, p_1 is a *proper prefix* of p, denoted

$p_1 < p$, if $p_1 \leq p$ and $p_1 \neq p$ hold. Two positions p, p' are parallel if $p \not\leq p'$ and $p' \not\leq p$ hold.

The *size* of a term t is denoted by $|t|$ and defined as $|\mathsf{Pos}(t)|$. The *subterm* of a term t at a position $p \in \mathsf{Pos}(t)$, denoted $t|_p$, is defined recursively as $t|_\lambda = t$ and $\alpha(t_1, \ldots, t_m)|_{i.p} = t_i|_p$. The *replacement* of a term t at a position $p \in \mathsf{Pos}(t)$ by a term s, denoted $t[s]_p$, is defined recursively as $t[s]_\lambda = s$ and $\alpha(t_1, \ldots, t_m)[s]_{i.p} = \alpha(t_1, \ldots, t_i[s]_p, \ldots, t_m)$. $\alpha(t_1, \ldots, t_m)$.

A *context* is a term with exactly one occurrence of a special constant symbol $\bullet$, called *hole*. The set of contexts over a ranked alphabet $\mathcal{F}$ and a set of variables $\mathcal{V}$ is denoted $\mathcal{C}(\mathcal{F}, \mathcal{V})$. We use c to denote a context in $\mathcal{C}(\mathcal{F}, \mathcal{V})$. The operations on terms defined above are extended to contexts in the natural way. The *hole position* of a context c, denoted $\mathsf{hp}(c)$, is the position in $\mathsf{Pos}(c)$ labeled by $\bullet$. Given a context c and a term t, we define the term $c[t]$ as $c[t]_{\mathsf{hp}(c)}$. Similarly, given another context c', the *concatenation* cc' is the context $c[c']_{\mathsf{hp}(c)}$. Note that $\mathsf{hp}(cc') = \mathsf{hp}(c).\mathsf{hp}(c')$ holds. The *exponentiation* of a context c to a natural number e, denoted c^e, is the context recursively defined as $c^e = cc^{e-|\mathsf{hp}(c)|}$ if $e > |\mathsf{hp}(c)| > 0$, as $c^e = c[\bullet]_p$ if $|\mathsf{hp}(c)| \geq e$, where $p \leq \mathsf{hp}(c)$ and $|p| = e$, and as $c^e = \bullet$ otherwise. Note that $|\mathsf{hp}(c^e)| = e$ holds for any context c with $|\mathsf{hp}(c)| > 0$ and natural number e. We say that c is a *subcontext* of a term t if $c = t|_{p_1}[\bullet]_{p_2}$, for $p_1.p_2 \in \mathsf{Pos}(t)$.

A *substitution*, denoted by σ, θ, is a total function $\sigma : \mathcal{V} \to \mathcal{T}(\mathcal{F}, \mathcal{V}) \cup \mathcal{C}(\mathcal{F}, \mathcal{V})$ such that $\sigma(\alpha) \in \mathcal{T}(\mathcal{F}, \mathcal{V})$ if α is a first-order variable and $\sigma(\alpha) \in \mathcal{C}(\mathcal{F}, \mathcal{V})$ if α is a context variable. The domain of a substitution σ is usually considered to be the set of variables α such that $\sigma(\alpha) \neq \alpha$, i.e. $\mathsf{Dom}(\sigma) = \{\alpha \mid \sigma(\alpha) \neq \alpha\}$. For this reason, when defining a particular substitution σ we do not make explicit $\sigma(\alpha)$ for variables $\alpha \notin \mathsf{Dom}(\sigma)$. We also define the variables occurring in a substitution σ as $\mathsf{Vars}(\sigma) = \bigcup_{\alpha \in \mathsf{Dom}(\sigma)} \{\alpha\} \cup \mathsf{Vars}(\sigma(\alpha))$. Moreover, substitutions are extended to be mappings from terms to terms, i.e. $\sigma : \mathcal{T}(\mathcal{F}, \mathcal{V}) \to \mathcal{T}(\mathcal{F}, \mathcal{V})$, as follows: $\sigma(g(t_1, \ldots, t_n)) = g(\sigma(t_1), \ldots, \sigma(t_n))$ and $\sigma(F(t)) = \sigma(F)[\sigma(t)]$. In addition, substitutions are also extended, in a similar way, to be mappings from contexts to contexts, i.e. $\sigma : \mathcal{C}(\mathcal{F}, \mathcal{V}) \to \mathcal{C}(\mathcal{F}, \mathcal{V})$. The *composition* of σ and θ, denoted $\theta \circ \sigma$, is defined as $\{\alpha \mapsto \theta(\sigma(\alpha)) \mid \alpha \in \mathsf{Dom}(\sigma) \cup \mathsf{Dom}(\theta)\}$.

2.2 Singleton Tree Grammars

A *Singleton Tree Grammar (STG)* is a 4-tuple $G = \langle \mathcal{TN}, \mathcal{CN}, \Sigma, R \rangle$, where $\mathcal{TN}$ is a set of nonterminals of arity 0, called *term nonterminals* and denoted by T, $\mathcal{CN}$ is a set of nonterminals of arity 1, called *context nonterminals* and denoted by C, and Σ is a ranked alphabet, whose elements are called *terminals*. The sets $\mathcal{TN}$, $\mathcal{CN}$, and Σ are pairwise disjoint. We denote by N nonterminals in $\mathcal{TN} \cup \mathcal{CN}$. R is a finite set of rules of the form: $T \to \alpha(T_1, \ldots, T_m)$, $T \to C_1 T_2$, $C \to \bullet$, $C \to C_1 C_2$, $C \to \alpha(T_1, \ldots, T_{i-1}, C_i, T_{i+1}, \ldots, T_m)$, $T \to T_1$, $C \to C_1$, where α is a terminal of arity m. STG's are nonrecursive, i.e. the transitive closure of the derivational relation between nonterminals is terminating. Furthermore, for every nonterminal N there is exactly one rule having N as left-hand side. Last two conditions guarantee that every nonterminal of an STG generates exactly one term/context. Given a term t (context c) with occurrences of nonterminals, the derivation of t (resp. c) by G is an exhaustive iterated replacement of the nonterminals by the corresponding right-hand sides. The result is denoted as $w_{G,t}$ (resp. $w_{G,c}$). In the case of a nonterminal N we also say that N *generates* $w_{G,N}$. The *size* of an STG G, denoted $|G|$, is the number of nonterminals of G. The size of the representation of an STG G is bounded by $|G| \cdot (2 + m)$, where m is the maximum arity of the terminals of G, but we use the other notion to ease the presentation.

Our definition of STG's is different from the one in [3], where nonterminals are allowed to generate contexts with several holes. Nevertheless, both notions were proven equivalent in [16] up to a polynomial transformation. In the present paper we use STG's for term

representation. It allows to represent terms with exponential size and height in linear space. This is in contrast with DAG's, which only allow for exponential compression in size.

▶ **Example 2.1.** Let n be a natural number. We define the STG G_n to have the following set of rules: $\{T \to C_n T_a,\ T_a \to a,\ C_\bullet \to \bullet,\ C_0 \to g(C_\bullet),\ C_1 \to C_0C_0,\ C_2 \to C_1C_1,\ C_3 \to C_2C_2,\ \ldots,\ C_n \to C_{n-1}C_{n-1}\}$. Note that $w_{G_n,T} = g^{2^n}(a)$, which would require exponential space both in an explicit and a DAG representation.

Several properties can be computed efficiently on STG's (see [5, 12]), e.g. computing $|w_{G,N}|$ for every nonterminal N of G or computing the symbol labeling a certain position of the generated term/context. A remarkable result used in the present paper is stated in the following lemma. The rest of known results on STG's needed in this work are presented in the following section.

▶ **Lemma 2.2** ([13, 3]). *Given an STG G and two term nonterminals T_1, T_2 of G, it is decidable in time $\mathcal{O}(|G|^3)$ whether $w_{G,T_1} = w_{G,T_2}$.*

2.3 Context Unification

Given two terms $s, t \in \mathcal{T}(\mathcal{F}, \mathcal{V})$, the *context unification* problem consists of deciding whether s and t are unifiable, i.e. whether there exists a substitution $\sigma : \mathcal{V} \to \mathcal{T}(\mathcal{F}, \mathcal{V}) \cup \mathcal{C}(\mathcal{F}, \mathcal{V})$ such that $\sigma(s) = \sigma(t)$. The *one-context unification* problem is the particular case of context unification in which $\mathcal{V}$ contains exactly one context variable. *First-order unification* corresponds to the particular case where there are no context variables in $\mathcal{V}$. For a fixed ranked alphabet $\mathcal{F}$, we define an *instance* of the one-context unification problem as a triple $\langle \Delta, \mathcal{X}, F \rangle$, where $\mathcal{X}$ is a set of first-order variables, F is a context variable, and Δ is a set of equations, i.e. a set of unordered pairs, of the form $s \doteq t$, with $s, t \in \mathcal{T}(\mathcal{F}, \mathcal{X} \cup \{F\})$. A solution of $\langle \Delta, \mathcal{X}, F \rangle$ is a substitution $\sigma : \mathcal{X} \cup \{F\} \to \mathcal{T}(\mathcal{F}, \mathcal{X} \cup \{F\}) \cup \mathcal{C}(\mathcal{F}, \mathcal{X} \cup \{F\})$ such that $\forall (s \doteq t) \in \Delta \ :\ \sigma(s) = \sigma(t)$. The definition of an instance that considers a single equation and ours are equivalent, but considering a set of equations is more appropriate in our setting.

▶ **Example 2.3.** Consider the set of equations $\{F(a) \doteq g(a, x), F(b) \doteq g(x, b)\}$. It can be unified by the substitution $\{F \mapsto g(\bullet, b), x \mapsto b\}$. Now consider the equation $F(g(x, b)) \doteq g(a, F(y))$. It has infinitely many solutions such as $\{F \mapsto g(a, \bullet), x \mapsto a, y \mapsto b\}$ and $\{F \mapsto g(a, g(a, \bullet)), x \mapsto a, y \mapsto b\}$. Finally, consider the set of equations $\{F(a) \doteq g(x, y), F(b) \doteq g(x, g(g(y, y), g(a, b)))\}$, which has no solution.

By considering that the input terms are compressed using STG's, we obtain *one-context unification with STG's*. Fixed a ranked alphabet $\mathcal{F}$, an *instance* of this problem is a tuple $\langle \Delta, G, \mathcal{X}, F \rangle$, where $\mathcal{X}$ is a set of first-order variables, F is a context variable, $G = \langle \mathcal{TN}, \mathcal{CN}, \mathcal{F} \cup \mathcal{X} \cup \{F\}, R \rangle$ is an STG, and Δ is a set of equations of the form $\{S_1 \doteq T_1, \ldots, S_n \doteq T_n\}$, where the S_i's and the T_i's are term nonterminals of G. With this representation, the context variable and first-order variables are initially represented as unary and constant terminal symbols of the grammar, respectively. Given an instance $\langle \{S_1 \doteq T_1, \ldots, S_n \doteq T_n\}, G, \mathcal{X}, F \rangle$, its corresponding uncompressed one-context unification instance is $\langle \{w_{G,S_1} \doteq w_{G,T_1}, \ldots, w_{G,S_n} \doteq w_{G,T_n}\}, \mathcal{X}, F \rangle$.

3 Known Results

3.1 Operations on STG's

In this section we describe the operations needed to compute a solution to an STG-compressed one-context unification instance. The following definition of *extension of an STG* describes the result of those operations.

▶ **Definition 3.1.** Let $\mathcal{F}$ be a ranked alphabet. Let $G = \langle \mathcal{TN}, \mathcal{CN}, \Sigma, R\rangle$ be an STG such that $\mathcal{F} \subseteq \Sigma$. An *$\mathcal{F}$-extension* of G is an STG $G' = \langle \mathcal{TN}', \mathcal{CN}', \Sigma', R'\rangle$ such that $\mathcal{TN} \subseteq \mathcal{TN}'$, $\mathcal{CN} \subseteq \mathcal{CN}'$, $R \subseteq R'$, and $\mathcal{F} \subseteq \Sigma'$.

The goal of the previous definition is to capture operations on STG's that correspond to instantiation of variables. More concretely, let $G = \langle \mathcal{TN}, \mathcal{CN}, \mathcal{F} \cup \mathcal{V}, R\rangle$ be an STG and let $G' = \langle \mathcal{TN}', \mathcal{CN}', \mathcal{F} \cup \mathcal{V}', R'\rangle$ be an $\mathcal{F}$-extension of G. The terms generated by term nonterminals of G and G' are assumed to belong to the sets $\mathcal{T}(\mathcal{F}, \mathcal{V})$ and $\mathcal{T}(\mathcal{F}, \mathcal{V}')$, respectively, and analogously for contexts. Hence, by defining a substitution $\sigma : \mathcal{V} \to \mathcal{T}(\mathcal{F}, \mathcal{V}') \cup \mathcal{C}(\mathcal{F}, \mathcal{V}')$ as $\sigma(\alpha) = w_{G',\alpha}$ it holds that, for each nonterminal N of G, $\sigma(w_{G,N}) = w_{G',N}$.

In [5] it is proven that first-order unification can be solved in polynomial time when the input terms are compressed using STG's. This approach proceeds analogously to the classical unification process. The main idea is that, given a first-order equation $s \doteq t$, the algorithm iteratively finds a position p labeled by a variable x in one of the terms, say s, and replaces all its occurrences by the corresponding subterm $t|_p$. This process ends when either s and t become equal, and thus they are unifiable, or a contradiction is reached. Since each iteration of the algorithm instantiates a variable, the solution σ can be described as an ordered sequence of substitutions on first-order variables. The authors focus on showing that this behaviour can be efficiently adapted to the STG-compressed setting. Besides checking equality at each iteration, the only operation that the algorithm needs to perform is to apply substitutions of the form $\{x \mapsto t|_p\}$. This corresponds to computing an $\mathcal{F}$-extension of the grammar that adds n new nonterminals $T_1, \ldots, T_n$ such that T_n generates $t|_p$ using $T_1, \ldots, T_{n-1}$ and adds a new rule $x \to T_n$, i.e. converting the terminal symbol x into a nonterminal generating $t|_p$. The crucial result to prove that such an $\mathcal{F}$-extension can be computed in polynomial time is that all such n are linearly bounded by the size of the initial input grammar and not by the size of the current grammar at each step of the process. Note that this implies that the size of the final grammar, i.e. the grammar obtained after all the variables have been replaced, is polynomially bounded by the size of the initial grammar. This technical fact is summarized in the following lemma, which is proven in [5].

▶ **Lemma 3.2.** *Let $G = \langle \mathcal{TN}, \mathcal{CN}, \Sigma, R\rangle$ be an STG describing a one-context unification instance. Let T be a term nonterminal of G and let t_0 be $w_{G,T}$. Let $t_1, \ldots, t_n$ be terms, $x_1, \ldots, x_n$ be first-order variables, $p_1, \ldots, p_n$ be positions, and $\sigma_1, \ldots, \sigma_n$ be substitutions satisfying, for $i \in \{1, \ldots, n\}$, $p_i \in \mathsf{Pos}(t_{i-1})$, $x_i \in \mathsf{Vars}(t_{i-1}) \setminus \mathsf{Vars}(t_{i-1}|_{p_i})$, $\sigma_i = \{x_i \mapsto t_{i-1}|_{p_i}\}$, and $t_i = \sigma_i(t_{i-1})$.*

Then, there exists an $\mathcal{F}$-extension $G' = \langle \mathcal{TN}', \mathcal{CN}', \Sigma \setminus \{x_1, \ldots, x_n\}, R'\rangle$ of G such that $w_{G',T} = t_n$ and $|G'| \leq |G| + n(|G| + 1)$.

In this paper we deal with one-context unification. Similar to the first-order case, instantiation of the special context variable F is performed by adding a rule $F \to C$ to the grammar, i.e. turning the terminal unary symbol F into a context nonterminal. This C is a new context nonterminal defined through the concatenation of several subcontexts of the input terms. In [5] it is shown how to efficiently compute subcontexts from a given STG-compressed term. The proposed construction guarantees that, given a grammar G, a nonterminal T and a position $p_1.p_2 \in \mathsf{Pos}(w_{G,T})$, G can be extended with, at most, $|G|(2|G| + 3)$ new nonterminals such that one of them generates the context $w_{G,T}|_{p_1}[\bullet]_{p_2}$. Moreover, given the context nonterminals $C_1, \ldots, C_n$ of a grammar G, an extended grammar containing a context nonterminal that generates the concatenation $w_{G,C_1} \ldots w_{G,C_n}$ can be easily obtained by adding n new nonterminals to G. These facts are stated in the following lemma, whose proof is also given in [5].

▶ **Lemma 3.3.** *Let $G = \langle \mathcal{TN}, \mathcal{CN}, \Sigma, R\rangle$ be an STG describing a one-context unification instance. Let T be a term nonterminal of G and let t be $w_{G,T}$. Let $p_1, \ldots, p_n, \hat{p}_1, \ldots, \hat{p}_n$ be positions and $c_1, \ldots, c_n$ be contexts satisfying, for $i \in \{1, \ldots, n\}$, $p_i.\hat{p}_i \in \mathsf{Pos}(t)$ and $c_i = t|_{p_i}[\bullet]_{\hat{p}_i}$.*

Then, there exists an $\mathcal{F}$-extension $G' = \langle \mathcal{TN}', \mathcal{CN}', \Sigma, R'\rangle$ of G with a context nonterminal C such that $w_{G',C} = c_1 \ldots c_n$ and $|G'| \leq |G| + n|G|(2|G| + 3) + n$.

In some cases, the context nonterminal C that instantiates F is defined using one last construction: context exponentiation. Computing the grammar that generates the result of this operation is straightforward, as shown in the following illustrative example.

▶ **Example 3.4.** Let G be an STG with the following set of rules: $\{T_g \to g(T_h), T_h \to h(T_a), T_a \to a\}$. Note that w_{G,T_g} is $g(h(a))$. The context exponentiation $(w_{G,T_g}|_\lambda[\bullet]_{1.1})^{11}$ is $g(h(g(h(g(h(g(h(g(h(g(\bullet)\ldots)$ and can be generated by the STG with the set of rules:

$$\begin{aligned} &\{C \to g(C_h), C_h \to h(C_\bullet), C_\bullet \to \bullet\} \\ \cup\ &\{C_{\mathsf{exp4}} \to CC, C_{\mathsf{exp8}} \to C_{\mathsf{exp4}}C_{\mathsf{exp4}}, C_{\mathsf{exp10}} \to C_{\mathsf{exp8}}C\} \\ \cup\ &\{C_{\mathsf{pref}} \to g(C_\bullet)\} \\ \cup\ &\{C_{\mathsf{exp11}} \to C_{\mathsf{exp10}}C_{\mathsf{pref}}\} \end{aligned}$$

Note that we use the nonterminals C and C_{pref} to generate two different subcontexts, $g(h(\bullet))$ and $g(\bullet)$, respectively. Moreover, the nonterminals C_{exp4}, C_{exp8}, C_{exp10}, and C_{exp11} are used for exponentiation and concatenation, with C_{exp11} generating the desired context.

As seen in the previous example, raising a context to a natural number e requires to (i) compute two different subcontexts c_1 and c_2, (ii) concatenate c_1 with itself several times, and (iii) concatenate the resulting context with c_2. The construction done in (i) adds, at most, $|G|(2|G| + 3)$ for each computed subcontext, (ii) can be performed efficiently because the number of new nonterminals to be added is logarithmic with respect to e, and (iii) only adds one extra nonterminal to the grammar. This fact is stated formally in the following lemma.

▶ **Lemma 3.5.** *Let $G = \langle \mathcal{TN}, \mathcal{CN}, \Sigma, R\rangle$ be an STG describing a one-context unification instance. Let T be a term nonterminal of G and let t be $w_{G,T}$. Let p_1, p_2 be positions such that $p_1.p_2 \in \mathsf{Pos}(t)$ and let $e \geq 0$ be a natural number.*

Then, there exists an $\mathcal{F}$-extension $G' = \langle \mathcal{TN}', \mathcal{CN}', \Sigma, R'\rangle$ of G with a context nonterminal C such that $w_{G',C} = (t|_{p_1}[\bullet]_{p_2})^e$ and $|G'| \leq |G| + 2|G|(2|G| + 3) + \lceil \log_2(e)\rceil + 1$.

3.2 Uncompressed One-Context Unification

Consider an instance $\langle \Delta, \mathcal{X}, F\rangle$ of the one-context unification problem. In [6] it is proven that this problem is in NP when the terms in Δ are represented explicitly. The proposed algorithm is presented as an inference system that modifies the set of equations until a contradiction is found or the empty set is derived, which implies unifiability. By representing Δ with DAG's, the size of the representation of the terms in the set of equations is guaranteed to stay polynomially bounded by the size of the input at each step of a derivation. For the sake of clarity and in order to make this paper self contained, we introduce in Figure 1 a simplified version of that inference system. In this simplified version we have erased the rules used to early detect nonunifiability and some restrictions that eased the proofs by guaranteeing a bound on the length of every derivation. Hence, the new version of the inference system is simpler and still sound. Moreover, since it is less restrictive than the original one, each derivation leading to a solution can still be performed, and thus it is complete.

The most basic rule of the inference system is the rule Decompose, which is just used to simplify the unification problem. Note that it does not introduce any new subterms in

$$\textsf{Decompose:}\quad \frac{\Delta = \Delta' \uplus \{\alpha(t_1,\dots,t_n) \doteq \alpha(u_1,\dots,u_n)\}}{\Delta' \cup \{t_1 \doteq u_1,\dots,t_n \doteq u_n\}}$$

$$\textsf{Var-Elim:}\quad \frac{x \doteq t \in \Delta}{\{x \mapsto t\}(\Delta)} \quad \text{where } x \notin \mathsf{Vars}(t)$$

$$\textsf{Var-Elim2:}\quad \frac{F(u) \doteq c[x] \in \Delta}{\{x \mapsto F(\theta(u))\}(\theta(\Delta))} \quad \text{where } \theta = \{F \mapsto c[F(\bullet)]\} \text{ and } c \text{ is guessed such that } F \notin \mathsf{Vars}(c),\ x \notin \mathsf{Vars}(u), \text{ and } (F \notin \mathsf{Vars}(u) \vee x \notin \mathsf{Vars}(c))$$

$$\textsf{CVar-Elim:}\quad \frac{F(u) \doteq c[t] \in \Delta}{\{F \mapsto c\}(\Delta)} \quad \text{where } c \text{ is guessed such that } F \notin \mathsf{Vars}(c)$$

$$\textsf{CVar-Elim2:}\quad \frac{F(u) \doteq c[F(t)] \in \Delta}{\{F \mapsto c^e\}(\Delta)} \quad \text{where } c \text{ is guessed such that } F \notin \mathsf{Vars}(c) \text{ and } e \text{ is guessed such that } 0 \le e \le 3\sum_{s \doteq t \in \Delta}(|s| + |t|)$$

Figure 1 Inference system for one-context unification.

the set of equations. The remaining rules modify the current set of equations by replacing variables by subterms and subcontexts constructed from the terms in the set of equations. In particular, rule Var-Elim replaces a first-order variable by a term, rule Var-Elim2 partially guesses the initial part of F and instantiates a first-order variable, and rules CVar-Elim and CVar-Elim2 replace F by a context. As a technical detail, note that the substitution applied due to the application of rule Var-Elim2 can be seen as an instantiation of F in terms of a freshly introduced context variable, which, for clarity, we denote also as F. Finally, the rule CVar-Elim2 instantiates the context variable by a context raised to a natural number whose value is linearly bounded by the size of the current set of equations. Note that it is a bound on the exponent of periodicity of minimal solutions, i.e. the maximum number of periodic repetitions of a context in a minimal solution.

▶ **Example 3.6.** Consider the one-context unification instance $\langle \Delta, \{x\}, F\rangle$, where Δ contains only the following equation: $F(g(a, F(b))) \doteq g(x, g(a, x))$.

This instance has no solution. Note that neither Decompose, Var-Elim, nor CVar-Elim2 can be applied. In the case of CVar-Elim, we need to choose a position in $\mathsf{Pos}(g(x, g(a, x)))$ in order to guess a context c. Hence, there exist five different options for c: $c = \bullet$, $c = g(\bullet, g(a, x))$, $c = g(x, \bullet)$, $c = g(x, g(\bullet, x))$, and $c = g(x, g(a, \bullet))$. Note that, in any case, the resulting first-order equation after applying the substitution $\{F \mapsto c\}$ to Δ does not lead to a solution. Finally, Var-Elim2 cannot be applied since the left-hand side of the equation is of the form $F(u)$ and F occurs in u, and the right-hand side of the equation has two occurrences of x. In order to understand the conditions of rule Var-Elim2, note that they allow either x to occur more than once in the right-hand side or F to occur in u. The reason to consider these situations separately is that both facts cannot hold at the same time since it would lead to an instantiation of F in terms of itself, and thus a contradiction.

By the form of the rules, the following statement bounding the length of the derivations holds trivially.

▶ **Lemma 3.7.** *Let $\langle \Delta, \mathcal{X}, F\rangle$ be a one-context unification instance. Any derivation from Δ using the inference system of Figure 1 contains at most $|\mathcal{X}|$ occurrences of* Var-Elim *and* Var-Elim2*, and at most one occurrence of either* CVar-Elim *or* CVar-Elim2*.*

4 Approach

In our setting, the terms in the initial set of equations Δ are represented by an STG. In order to prove that one-context unification is also in NP in the STG-compressed case, we adapt the inference system described in Section 3.2. Since subterms, subcontexts, context exponentiation, and variable instantiations are known to be efficiently computable with STG's, as seen in Section 3.1, one may be tempted to simply reproduce the sequence of variable instantiations done in the uncompressed case. However, the size of the grammar may grow after certain operations and, in order to prove that it does not explode, we need to use a different approach. In particular, the difficulties rely on the fact that we do not have the analogous of Lemma 3.2 for successive partial instantiations of the context variable, which require to compute a subcontext and thus increase the size of the grammar.

Our approach consists of modifying the sequences of rule applications that describe a solution in order to guarantee that they can be represented in polynomial space using an STG. To simplify reasonings, we introduce in Figure 2 a new inference system $\mathfrak{R}$ that generalizes the previous one in Figure 1. In $\mathfrak{R}$ we assume without loss of generality that the initial set of equations $\Delta = \{s_1 \doteq t_1, \ldots, s_n \doteq t_n\}$ is encoded as a single term. This can be done by extending the alphabet with new symbols $\mathfrak{d}$ and $\mathfrak{e}$ of arity n and 2, respectively, and defining $\mathsf{term}(\Delta) = \mathfrak{d}(\mathfrak{e}(s_1, t_1), \ldots, \mathfrak{e}(s_n, t_n))$. With this notion, the question of whether there exists a substitution σ, the solution for Δ, such that $\sigma(s_1) = \sigma(t_1), \ldots, \sigma(s_n) = \sigma(t_n)$ corresponds to check whether there exists a substitution σ, the solution for $\mathsf{term}(\Delta)$, such that $\sigma(\mathsf{term}(\Delta))$ is of the form $\mathfrak{d}(\mathfrak{e}(u_1, u_1), \ldots, \mathfrak{e}(u_n, u_n))$. This change in notation is useful to refer to subterms of both sides of the equations in Δ indistinctly as subterms of $\mathsf{term}(\Delta)$.

$$\mathrm{R_x}: \quad \frac{t}{\{x \mapsto t|_p\}(t)} \quad \text{where } p \in \mathsf{Pos}(t) \text{ and } x \in \mathsf{Vars}(t) \setminus \mathsf{Vars}(t|_p)$$

$$\mathrm{R_{FF}}: \quad \frac{t}{\{F \mapsto t|_{p_1}[F(\bullet)]_{p_2}\}(t)} \quad \text{where } p_1.p_2 \in \mathsf{Pos}(t) \text{ and } F \in \mathsf{Vars}(t) \setminus \mathsf{Vars}(t|_{p_1})$$

$$\mathrm{R_{FC}}: \quad \frac{t}{\{F \mapsto (t|_{p_1}[\bullet]_{p_2})^e\}(t)} \quad \text{where } p_1.p_2 \in \mathsf{Pos}(t),\ F \in \mathsf{Vars}(t) \setminus \mathsf{Vars}(t|_{p_1}[\bullet]_{p_2}),\ \text{and } e \in \{0, \ldots, 3|t|\}$$

Figure 2 The adapted inference system $\mathfrak{R}$.

It is easy to see that an application of Var-Elim, Var-Elim2, CVar-Elim, or CVar-Elim2 corresponds to the application of at most two of the rules of $\mathfrak{R}$. In particular, an application of Var-Elim is emulated by an application of $\mathrm{R_x}$. In the rest of this paper, we use $\mathrm{R_{xF}}$ and $\mathrm{R_{x\neg F}}$ to refer to applications of $\mathrm{R_x}$ in which the involved subterm $t|_p$ has an occurrence of F or not, respectively. An application of Var-Elim2 corresponds to an application of $\mathrm{R_{FF}}$ followed by an application of $\mathrm{R_{xF}}$. Finally, applications of CVar-Elim and CVar-Elim2 are emulated by $\mathrm{R_{FC}}$. The remaining original rule, Decompose, was only used to simplify the problem in order to apply other rules. Its behaviour is implicitly emulated in $\mathfrak{R}$ by defining its rules by means of subterms and subcontexts of t.

With $\to_{\mathrm{R_{xF}},x,p}$ and $\to_{\mathrm{R_{x\neg F}},x,p}$ we denote an application of rules $\mathrm{R_{xF}}$ and $\mathrm{R_{x\neg F}}$, respectively, making explicit the position p and the first-order variable x involved in the rule application. Analogously, with $\to_{\mathrm{R_{FF}},p_1,p_2}$ and $\to_{\mathrm{R_{FC}},p_1,p_2}$ we denote an application of $\mathrm{R_{FF}}$ and $\mathrm{R_{FC}}$, making explicit the positions p_1 and p_2 involved in the rule application. Sometimes, we do not make explicit x, p, p_1, and p_2 when they are clear from the context or not relevant. By $\to_{\mathfrak{R}}$ we denote the derivational relation using $\mathfrak{R}$ and, as usual, $\to^+_{\mathfrak{R}}$ denotes its transitive closure and $\to^*_{\mathfrak{R}}$ denotes its reflexive-transitive closure. Additionally, by $\to_r$ and $\to^*_r$, we

denote the derivational relation using the rule r of $\mathfrak{R}$ and its reflexive-transitive closure, respectively.

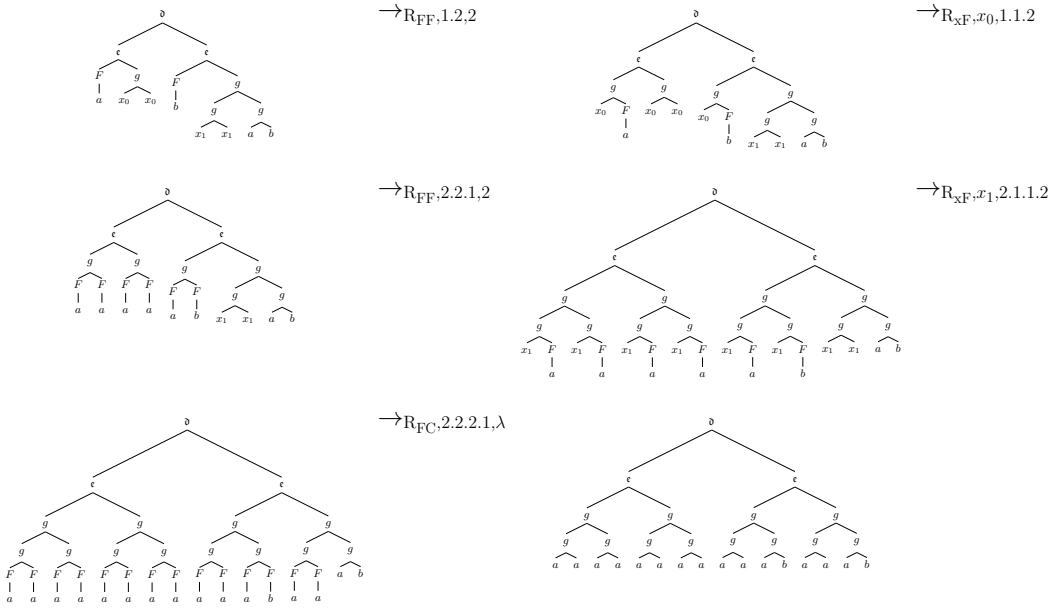
The following example illustrates the fact that $\mathfrak{R}$ can emulate derivations done with the inference system in Figure 1.

▶ **Example 4.1.** Consider the one-context unification instance $\langle\{F(a) \doteq g(x_0, x_0),\ F(b) \doteq g(g(x_1, x_1), g(a, b))\}, \{x_0, x_1\}, F\rangle$. In this example we use $\to_r^\sigma$ to denote a derivation step using the rule r and applying the substitution σ and $\to^*_{\mathsf{Decompose}}$ to denote some derivation steps using the rule Decompose. A possible successful derivation using the rules of Figure 1 is the following:

$$
\begin{aligned}
&\{F(a) \doteq g(x_0, x_0),\ F(b) \doteq g(g(x_1, x_1), g(a, b))\}\\
\to^{\{x_0 \mapsto F(\{F \mapsto g(x_0, F(\bullet))\}(a))\} \circ \{F \mapsto g(x_0, F(\bullet))\}}_{\mathsf{Var-Elim2}}\\
&\{g(F(a), F(a)) \doteq g(F(a), F(a)),\ g(F(a), F(b)) \doteq g(g(x_1, x_1), g(a, b))\}\\
\to^*_{\mathsf{Decompose}}\\
&\{F(a) \doteq g(x_1, x_1),\ F(b) \doteq g(a, b)\}\\
\to^{\{x_1 \mapsto F(\{F \mapsto g(x_1, F(\bullet))\}(a))\} \circ \{F \mapsto g(x_1, F(\bullet))\}}_{\mathsf{Var-Elim2}}\\
&\{g(F(a), F(a)) \doteq g(F(a), F(a)),\ g(F(a), F(b)) \doteq g(a, b)\}\\
\to^*_{\mathsf{Decompose}}\ &\{F(a) \doteq a,\ F(b) \doteq b\}\ \to^{\{F \mapsto \bullet\}}_{\mathsf{CVar-Elim}}\ \{a \doteq a,\ b \doteq b\}\ \to^*_{\mathsf{Decompose}}\ \emptyset
\end{aligned}
$$

Since we could derive $\emptyset$, the considered instance is indeed unifiable. The solution associated to the derivation is $\{F \mapsto g(g(a, a), g(a, \bullet)), x_0 \mapsto g(a, a), x_1 \mapsto a\}$.

We now show that the same solution can be derived also using $\mathfrak{R}$. The initial set of equations, expressed with the new notation, corresponds to the term $t = \mathsf{term}(\Delta) = \mathfrak{d}(\mathfrak{e}(F(a), g(x_0, x_0)), \mathfrak{e}(F(b), g(g(x_1, x_1), g(a, b))))$. Note that the subterm $t|_1$ encodes the equation $F(a) \doteq g(x_0, x_0)$ and $t|_2$ encodes $F(b) \doteq g(g(x_1, x_1), g(a, b))$. Our goal is to check whether there exists a substitution σ such that $\sigma(t)$ is of the form $\mathfrak{d}(\mathfrak{e}(t_1, t_1), \mathfrak{e}(t_2, t_2))$. The previous derivation corresponds to the following one using $\mathfrak{R}$:



Note that, there are several equivalent options for the selection of the first position used in the last rule application. We chose 2.2.2.1 because this is the position that corresponds to the last steps of the previous derivation.

It is trivial that $\mathfrak{R}$ is sound. Moreover, since $\mathfrak{R}$ can emulate the original inference system, it follows that it is also complete. The following lemma states that soundness and completeness of $\mathfrak{R}$ also hold when the length of derivations is linearly bounded. This property follows from completeness of the inference system of Figure 1, soundness of $\mathfrak{R}$, and Lemma 3.7, taking into account that each rule of Figure 1 can be emulated with at most two rules of $\mathfrak{R}$.

▶ **Lemma 4.2.** *Let $\langle \Delta, \mathcal{X}, F\rangle$ be a one-context unification instance. Δ has a solution if and only if there exists a derivation $\mathsf{term}(\Delta) \to^*_{\mathfrak{R}} t$ of length at most $2|\mathcal{X}| + 1$ such that t is of the form $\mathfrak{d}(\mathfrak{e}(u_1, u_1), \ldots, \mathfrak{e}(u_{|\Delta|}, u_{|\Delta|}))$.*

In the case of one-context unification with STG's, $\mathsf{term}(\Delta)$ is represented by a single term nonterminal. More concretely, an instance of one-context unification with STG's $\langle \{S_1 \doteq T_1, \ldots, S_n \doteq T_n\}, \bar{G}, \mathcal{X}, F\rangle$ is represented as $\langle T, G, \mathcal{X}, F\rangle$, where G is an $\mathcal{F}$-extension of $\bar{G}$ and T is a term nonterminal of G such that $w_{G,T} = \mathfrak{d}(\mathfrak{e}(w_{\bar{G},S_1}, w_{\bar{G},T_1}), \ldots, \mathfrak{e}(w_{\bar{G},S_n}, w_{\bar{G},T_n})) = \mathsf{term}(\{w_{\bar{G},S_1} \doteq w_{\bar{G},T_1}, \ldots, w_{\bar{G},S_n} \doteq w_{\bar{G},T_n}\}) = \mathsf{term}(\Delta)$. With this new representation, the problem consists of deciding whether there exists a substitution σ such that $\sigma(w_{G,T})$ is of the form $\mathfrak{d}(\mathfrak{e}(u_1, u_1), \ldots, \mathfrak{e}(u_n, u_n))$. Recall that NP can be defined as the set of decisional problems whose positive instances can be verified in polynomial time. Hence, to prove that one-context unification with STG's is in NP we need to prove that there exists an $\mathcal{F}$-extension G' of G of polynomial size with respect to $|G|$ that represents σ. More concretely, in the rest of the paper we prove that, for each derivation of the form $\mathsf{term}(\Delta) \to^*_{\mathfrak{R}} t$ of length at most $2|\mathcal{X}| + 1$, there exists an $\mathcal{F}$-extension G' of G such that $w_{G',T} = t$ and whose size is polynomial with respect to $|G|$. This is enough for proving that one-context unification with STG's is in NP since the fact that t is of the form $\mathfrak{d}(\mathfrak{e}(u_1, u_1), \ldots, \mathfrak{e}(u_n, u_n))$ can be checked in polynomial time with respect to $|G'|$.

5 Commutation of Substitutions

As commented in the previous section, our approach consists of modifying the sequences of derivation steps with $\mathfrak{R}$ in such a way that allows to conclude that every unifiable instance $\langle \Delta, \mathcal{X}, F\rangle$ has a solution σ that can be represented in polynomial space using an STG. Our goal is to show that rule applications can always be commuted to obtain an equivalent derivation, i.e. a derivation describing the same substitution σ, that is of the following form:

$$\mathsf{term}(\Delta) \to^*_{\mathrm{R}_{\mathrm{x}\neg\mathrm{F}}} \to^*_{\mathrm{R}_{\mathrm{FF}}} \to^*_{\mathrm{R}_{\mathrm{xF}}} \to^{0,1}_{\mathrm{R}_{\mathrm{FC}}} \to^*_{\mathrm{R}_{\mathrm{x}\neg\mathrm{F}}} \sigma(\mathsf{term}(\Delta))$$

where $\to^{0,1}_{\mathrm{R}_{\mathrm{FC}}}$ denotes either 0 or 1 applications of the rule R_{FC}. As a first ingredient in this argument, we define a particular notion of composition of substitutions which will be useful to reason in our setting. Next, we present some technical results to, finally, prove how to commute derivation steps until obtaining an equivalent derivation of the desired form.

5.1 Strong Composition

Consider two substitutions σ_1, σ_2. The goal of the following notion of composition of σ_1 and σ_2 is to capture how instantiations due to σ_1 are modified by the later application of σ_2.

▶ **Definition 5.1.** Let σ_1, σ_2 be substitutions. The *strong composition* of σ_1 and σ_2, denoted $\sigma_2 \diamond \sigma_1$, is defined as $\{\alpha \mapsto \sigma_2(\sigma_1(\alpha)) \mid \alpha \in \mathsf{Dom}(\sigma_1)\}$

The usual and the strong notions of composition are not equivalent in general. Recall that, given substitutions σ_1, σ_2, the usual notion of composition can be defined as $\sigma_2 \circ \sigma_1 =$

$\{\alpha \mapsto \sigma_2(\sigma_1(\alpha)) \mid \alpha \in \mathsf{Dom}(\sigma_1) \cup \mathsf{Dom}(\sigma_2)\}$. In order to stress the difference, consider $\theta_1 = \{y \mapsto b\}$ and $\theta_2 = \{x \mapsto a\}$ and note that $(\theta_2 \diamond \theta_1)(x) = x$, while $(\theta_2 \circ \theta_1)(x) = a$. Moreover, strong composition is not associative, i.e. $(\sigma_3 \diamond \sigma_2) \diamond \sigma_1 = \sigma_3 \diamond (\sigma_2 \diamond \sigma_1)$ does not hold in general: consider $\theta_0 = \{y \mapsto g(x)\}$ and note that $(\theta_2 \diamond \theta_1) \diamond \theta_0 = \{y \mapsto g(x)\}$, while $\theta_2 \diamond (\theta_1 \diamond \theta_0) = \{y \mapsto g(a)\}$. Another property that distinguishes both notions of composition is that, when using strong composition, $\mathsf{Dom}(\sigma_2 \diamond \sigma_1) \subseteq \mathsf{Dom}(\sigma_1)$ holds. This inclusion is strict only in anomalous cases, for example $\mathsf{Dom}(\{y \mapsto x\} \diamond \{x \mapsto y\}) = \emptyset$. In fact, the condition $\mathsf{Dom}(\sigma_2 \diamond \sigma_1) = \mathsf{Dom}(\sigma_1)$ is ensured when $\mathsf{Vars}(\sigma_2) \cap \mathsf{Dom}(\sigma_1) = \emptyset$, which is usually the case in our setting.

The following lemma is straightforward from the definition of strong composition.

▶ **Lemma 5.2.** *Let σ_1, σ_2 be substitutions and let α be a variable in $\mathsf{Dom}(\sigma_1)$. Then, $(\sigma_2 \circ \sigma_1)(\alpha) = (\sigma_2 \diamond \sigma_1)(\alpha)$.*

The following lemma states how two substitutions can be "commuted" using the strong composition.

▶ **Lemma 5.3.** *Let σ_1, σ_2 be substitutions such that $\mathsf{Vars}(\sigma_2) \cap \mathsf{Dom}(\sigma_1) = \emptyset$. Then, $\sigma_2 \circ \sigma_1 = (\sigma_2 \diamond \sigma_1) \circ \sigma_2$.*

Proof. Let $\mathcal{V}$ be a set of variables such that $\mathsf{Dom}(\sigma_1) \cup \mathsf{Dom}(\sigma_2) \subseteq \mathcal{V}$ holds. It suffices to prove that, for all $\alpha \in \mathcal{V}$, $(\sigma_2 \circ \sigma_1)(\alpha) = ((\sigma_2 \diamond \sigma_1) \circ \sigma_2)(\alpha)$ holds.

We distinguish cases depending on whether $\alpha \in \mathsf{Dom}(\sigma_1)$ and $\alpha \in \mathsf{Dom}(\sigma_2)$. If $\alpha \in \mathsf{Dom}(\sigma_1)$ then, by the assumption of the lemma, $\alpha \notin \mathsf{Dom}(\sigma_2)$ holds, and hence $((\sigma_2 \diamond \sigma_1) \circ \sigma_2)(\alpha) = (\sigma_2 \diamond \sigma_1)(\alpha) = (\sigma_2 \circ \sigma_1)(\alpha)$ holds by Lemma 5.2. If $\alpha \in \mathsf{Dom}(\sigma_2)$ then, by the assumption of the lemma, $\mathsf{Vars}(\sigma_2(\alpha)) \cap \mathsf{Dom}(\sigma_2 \diamond \sigma_1) = \emptyset$ and $\alpha \notin \mathsf{Dom}(\sigma_1)$ hold, and it follows $((\sigma_2 \diamond \sigma_1) \circ \sigma_2)(\alpha) = \sigma_2(\alpha) = (\sigma_2 \circ \sigma_1)(\alpha)$. Finally, the case where $\alpha \notin \mathsf{Dom}(\sigma_1) \cup \mathsf{Dom}(\sigma_2)$ trivially holds and the case where $\alpha \in \mathsf{Dom}(\sigma_1) \cap \mathsf{Dom}(\sigma_2)$ is not possible by the assumption. ◀

5.2 Subterm Preservation

Let t_1 and t_2 be terms such that t_2 is obtained from t_1 by applying a substitution. The following two lemmas state under which conditions a certain subterm of t_2 exists also as a subterm of t_1. These results will be crucial to argue about the commutation of derivation steps.

▶ **Lemma 5.4.** *Let $\mathcal{F}$ be a ranked alphabet and let $\mathcal{V}$ be a set with first-order variables and a context variable F. Let t_1, t_2 be terms in $\mathcal{T}(\mathcal{F}, \mathcal{V})$ such that $t_2 = \{F \mapsto c[F(\bullet)]\}(t_1)$, where c is a context in $\mathcal{C}(\mathcal{F}, \mathcal{V})$. Let p be a position in $\mathsf{Pos}(t_2)$ such that $F \notin \mathsf{Vars}(t_2|_p)$. Then, either there exists a position $\hat{p} \in \mathsf{Pos}(t_1)$ such that $t_1|_{\hat{p}} = t_2|_p$ or there exists a position $\hat{p} \in \mathsf{Pos}(c)$ such that $c|_{\hat{p}} = t_2|_p$.*

Proof. We prove the lemma by induction on $|p|$. If $p = \lambda$, note that F cannot occur in t_1 since, otherwise, $F \in \mathsf{Vars}(t_2|_p)$. Hence, in this case, $t_1 = t_2$ holds and we are done by defining $\hat{p}$ as p. For the induction step, assume that $|p| > 0$ and consider the following cases.

First, consider that t_1 is of the form $g(u_1, \ldots, u_n)$, where g is a function symbol in $\mathcal{F}$. Note that $n > 0$ holds since, otherwise, $t_2 = g$ and $p = \lambda$, contradicting the assumption. Let p be $i.p'$ more explicitly written, for $i \in \{1, \ldots, n\}$. Note that $t_2|_i = \{F \mapsto c[F(\bullet)]\}(u_i)$, $p' \in \mathsf{Pos}(t_2|_i)$, and $F \notin \mathsf{Vars}(t_2|_i|_{p'})$ hold. By induction hypothesis on $|p'|$, there exists a position $\hat{p}'$ such that either $u_i|_{\hat{p}'} = t_2|_i|_{p'}$ or $c|_{\hat{p}'} = t_2|_i|_{p'}$. The statement follows by defining $\hat{p}$ as $i.\hat{p}'$ in the former case and as $\hat{p}'$ in the latter case.

Second, consider that t_1 is of the form $F(u)$. We distinguish cases depending on whether p and $\mathsf{hp}(c)$ are parallel or not. Note that p cannot be a prefix of $\mathsf{hp}(c)$ since it would lead to a contradiction with the fact that $F \notin \mathsf{Vars}(t_2|_p)$. In the case where p and $\mathsf{hp}(c)$ are parallel, it follows that $c|_p = t_2|_p$, and we are done by defining $\hat{p}$ as p. Otherwise, let p be $\mathsf{hp}(c).1.p'$ more explicitly written. In this case, note that $t_2|_{\mathsf{hp}(c).1} = \{F \mapsto c[F(\bullet)]\}(u)$, $p' \in \mathsf{Pos}(t_2|_{\mathsf{hp}(c).1})$, and $F \notin \mathsf{Vars}(t_2|_{\mathsf{hp}(c).1}|_{p'})$ hold. Hence, by induction hypothesis on $|p'|$, there exists a position $\hat{p}'$ such that either $u|_{\hat{p}'} = t_2|_{\mathsf{hp}(c).1}|_{p'}$ or $c|_{\hat{p}'} = t_2|_{\mathsf{hp}(c).1}|_{p'}$. The statement follows by defining $\hat{p}$ as $1.\hat{p}'$ in the former case and as $\hat{p}'$ in the latter case.

Third, the case where t_1 is a first-order variable is not possible because, in such case, p must be λ, which contradicts the assumption. ◀

▶ **Lemma 5.5.** *Let $\mathcal{F}$ be a ranked alphabet and let $\mathcal{V}$ be a set with first-order variables and a context variable F. Let t_1, t_2 be terms in $\mathcal{T}(\mathcal{F}, \mathcal{V})$ such that either $t_1 \to_{\mathrm{R_{xF}}} t_2$ or $t_1 \to_{\mathrm{R_{FF}}} t_2$. Let p be a position in $\mathsf{Pos}(t_2)$ such that $F \notin \mathsf{Vars}(t_2|_p)$. Then, there exists a position $\hat{p} \in \mathsf{Pos}(t_1)$ such that $t_1|_{\hat{p}} = t_2|_p$.*

Proof. We first consider the case where the applied rule is $\mathrm{R_{FF}}$. Let $t_1 \to_{\mathrm{R_{FF}},p_1,p_2} t_2$ be the derivation step of the statement more explicitly written. Hence, $t_2 = \{F \mapsto t_1|_{p_1}[F(\bullet)]_{p_2}\}(t_1)$ and, by Lemma 5.4, the statement holds.

Now assume that the applied rule is $\mathrm{R_{xF}}$ with variable x and position q. Let P be the subset of positions of $\mathsf{Pos}(t_1)$ labeled by x in t_1. Note that P is a set of pairwise parallel positions. Moreover, note that p cannot be a prefix of any of the positions in P since, otherwise, $F \in \mathsf{Vars}(t_2|_p)$, contradicting the assumptions of the lemma. In the case where p is parallel with every position in P, it follows that $p \in \mathsf{Pos}(t_1)$ and $t_1|_p = t_2|_p$. Otherwise, exactly one position $p' \in P$ is a proper prefix of p. Hence, p is of the form $p'.q'$ and it follows that $t_1|_{q.q'} = t_2|_p$. ◀

The following lemma states how the instantiation of a context variable and the computation of a subterm can be commuted.

▶ **Lemma 5.6.** *Let $\mathcal{F}$ be a ranked alphabet and let $\mathcal{V}$ be a set with first-order variables and a context variable F. Let t be a term in $\mathcal{T}(\mathcal{F}, \mathcal{V})$, let p be a position in $\mathsf{Pos}(t)$, and let $\sigma = \{F \mapsto c\}$ be a substitution, where c is a context in $\mathcal{C}(\mathcal{F}, \mathcal{V})$. Then, there exists a position $\hat{p} \in \mathsf{Pos}(\sigma(t))$ such that $\sigma(t|_p) = \sigma(t)|_{\hat{p}}$.*

Proof. We prove the lemma by induction on $|p|$. The base case, i.e. when $p = \lambda$, trivially holds by defining $\hat{p}$ as p. For the induction step, assume that $|p| > 0$. We distinguish cases depending on the form of t.

First, assume that t is of the form $g(u_1, \ldots, u_n)$, where g is a function symbol in $\mathcal{F}$. Note that $n > 0$ necessarily holds since, otherwise, $p = \lambda$, contradicting the assumption. Let p be $i.p'$ more explicitly written, for $i \in \{1, \ldots, n\}$. By induction hypothesis, there exists a position $\hat{p}' \in \mathsf{Pos}(\sigma(u_i))$ such that $\sigma(u_i|_{p'}) = \sigma(u_i)|_{\hat{p}'}$ holds. Hence, the statement holds by defining $\hat{p}$ as $i.\hat{p}'$.

Second, assume that t is of the form $F(u)$. Let p be $1.p'$ more explicitly written. By induction hypothesis, there exists a position $\hat{p}' \in \mathsf{Pos}(\sigma(u))$ such that $\sigma(u|_{p'}) = \sigma(u)|_{\hat{p}'}$ holds. Hence, the statement holds by defining $\hat{p}$ as $\mathsf{hp}(c).\hat{p}'$.

Third, the case where t is a first-order variable is not possible because, in such case, p must be λ, which contradicts the assumption. ◀

5.3 Reordering Derivations

In this section we show how derivations with $\mathfrak{R}$ can be modified in order to guarantee that rules are applied in a specific order. As basic ingredients, the following three technical lemmas show how pairs of derivation steps can be swapped.

▶ **Lemma 5.7.** *Let $\mathcal{F}$ be a ranked alphabet and let $\mathcal{V}$ be a set with first-order variables and a context variable F. Let t_1, t_2, t be terms in $\mathcal{T}(\mathcal{F}, \mathcal{V})$ such that $t_1 \to_{\mathrm{R_{xF}},x,p} t \to_{\mathrm{R_{FF}},p_1,p_2} t_2$. Then, there exists a position $\hat{p}_1 \in \mathsf{Pos}(t_1)$, a term $t' \in \mathcal{T}(\mathcal{F}, \mathcal{V})$, and a position $\hat{p} \in \mathsf{Pos}(t')$ such that $t_1 \to_{\mathrm{R_{FF}},\hat{p}_1,p_2} t' \to_{\mathrm{R_{xF}},x,\hat{p}} t_2$ holds.*

Proof. By the conditions on the application of $\mathrm{R_{xF}}$, $x \in \mathsf{Vars}(t_1) \setminus \mathsf{Vars}(t_1|_p)$ and $F \in \mathsf{Vars}(t_1|_p)$ hold. In addition, since $\mathrm{R_{xF}}$ instantiates x, then $x \notin \mathsf{Vars}(t)$ holds. Moreover, by the conditions on the application of $\mathrm{R_{FF}}$, we know that $F \in \mathsf{Vars}(t) \setminus \mathsf{Vars}(t|_{p_1})$. Note that

$$
\begin{aligned}
t_2 &= \{F \mapsto t|_{p_1}[F(\bullet)]_{p_2}\}(\{x \mapsto t_1|_p\}(t_1)) && (1)\\
&= (\{F \mapsto t|_{p_1}[F(\bullet)]_{p_2}\} \diamond \{x \mapsto t_1|_p\})(\{F \mapsto t|_{p_1}[F(\bullet)]_{p_2}\}(t_1)) && (2)\\
&= \{x \mapsto \{F \mapsto t|_{p_1}[F(\bullet)]_{p_2}\}(t_1|_p)\}(\{F \mapsto t|_{p_1}[F(\bullet)]_{p_2}\}(t_1)) && (3)\\
&= \{x \mapsto \{F \mapsto t_1|_{\hat{p}_1}[F(\bullet)]_{p_2}\}(t_1|_p)\}(\{F \mapsto t_1|_{\hat{p}_1}[F(\bullet)]_{p_2}\}(t_1)) && (4)\\
&= \{x \mapsto \{F \mapsto t_1|_{\hat{p}_1}[F(\bullet)]_{p_2}\}(t_1)|_{\hat{p}}\}(\{F \mapsto t_1|_{\hat{p}_1}[F(\bullet)]_{p_2}\}(t_1)) && (5)
\end{aligned}
$$

where (1) follows from definition of $\mathrm{R_{xF}}$ and $\mathrm{R_{FF}}$, (2) follows from Lemma 5.3, which can be applied because $\mathsf{Vars}(\{F \mapsto t|_{p_1}[F(\bullet)]_{p_2}\}) \cap \mathsf{Dom}(\{x \mapsto t_1|_p\}) = \emptyset$ holds since $x \notin \mathsf{Vars}(t)$ and $x \neq F$, (3) follows from Definition 5.1, in (4) the implicit definition of $\hat{p}_1$ holding $t_1|_{\hat{p}_1} = t|_{p_1}$ follows from Lemma 5.5, which can be applied since $t_1 \to_{\mathrm{R_{xF}},x,p} t$ and $F \notin \mathsf{Vars}(t|_{p_1})$, and in (5) the implicit definition of $\hat{p}$ holding $\{F \mapsto t_1|_{\hat{p}_1}[F(\bullet)]_{p_2}\}(t_1)|_{\hat{p}} = \{F \mapsto t_1|_{\hat{p}_1}[F(\bullet)]_{p_2}\}(t_1|_p)$ follows from Lemma 5.6.

Finally, let the t' of the lemma be defined as $\{F \mapsto t_1|_{\hat{p}_1}[F(\bullet)]_{p_2}\}(t_1)$ and note that $t_2 = \{x \mapsto t'|_{\hat{p}}\}(t')$. It remains to prove that t' can be derived from t_1, and that t_2 can be derived from t', as stated in the lemma. First, note that $t_1 \to_{\mathrm{R_{FF}},\hat{p}_1,p_2} t'$ holds because F does not occur in $t_1|_{\hat{p}_1}$, since $t_1|_{\hat{p}_1} = t|_{p_1}$, and, moreover, $F \in \mathsf{Vars}(t_1)$ holds because $F \in \mathsf{Vars}(t)$ and $\mathsf{Vars}(t) \subsetneq \mathsf{Vars}(t_1)$. Now, note that $t' \to_{\mathrm{R_{xF}},x,\hat{p}} t_2$ holds because:

- $x \in \mathsf{Vars}(t')$, which follows from the facts that $x \in \mathsf{Vars}(t_1)$ and $\mathsf{Vars}(t') = \mathsf{Vars}(t_1)$,
- $x \notin \mathsf{Vars}(t'|_{\hat{p}})$, since $x \notin \mathsf{Vars}(t_1|_p)$, $x \notin \mathsf{Vars}(t)$, and $t'|_{\hat{p}} = \{F \mapsto t_1|_{\hat{p}_1}[F(\bullet)]_{p_2}\}(t_1)|_{\hat{p}} = \{F \mapsto t|_{p_1}[F(\bullet)]_{p_2}\}(t_1|_p)$, and
- $F \in \mathsf{Vars}(t'|_{\hat{p}})$, since $F \in \mathsf{Vars}(t_1|_p)$ holds and thus $F \in \mathsf{Vars}(\{F \mapsto t_1|_{\hat{p}_1}[F(\bullet)]_{p_2}\}(t_1|_p)) = \mathsf{Vars}(\{F \mapsto t_1|_{\hat{p}_1}[F(\bullet)]_{p_2}\}(t_1)|_{\hat{p}}) = \mathsf{Vars}(t'|_{\hat{p}})$ also holds.

Hence, $t_1 \to_{\mathrm{R_{FF}},\hat{p}_1,p_2} t' \to_{\mathrm{R_{xF}},x,\hat{p}} t_2$ holds, which concludes the proof. ◀

The proofs of the following two lemmas are very similar and, due to lack of space, the second one is omitted.

▶ **Lemma 5.8.** *Let $\mathcal{F}$ be a ranked alphabet and let $\mathcal{V}$ be a set with first-order variables and a context variable F. Let t_1, t_2, t be terms in $\mathcal{T}(\mathcal{F}, \mathcal{V})$ such that $t_1 \to_{\mathrm{R_{FF}},p_1,p_2} t \to_{\mathrm{R_{x\neg F}},x,p} t_2$. Then, there exists a position $\hat{p} \in \mathsf{Pos}(t_1)$ and a term $t' \in \mathcal{T}(\mathcal{F}, \mathcal{V})$ such that $t_1 \to_{\mathrm{R_{x\neg F}},x,\hat{p}} t' \to_{\mathrm{R_{FF}},p_1,p_2} t_2$ holds.*

Proof. By the conditions on the application of rule $\mathrm{R_{FF}}$, $F \in \mathsf{Vars}(t_1) \setminus \mathsf{Vars}(t_1|_{p_1})$ holds. Moreover, by the conditions on the application of rule $\mathrm{R_{x\neg F}}$, we know that

$x \in \mathsf{Vars}(t) \setminus \mathsf{Vars}(t|_p)$ and $F \notin \mathsf{Vars}(t|_p)$. Note that

$$
\begin{aligned}
t_2 =& \{x \mapsto t|_p\}(\{F \mapsto t_1|_{p_1}[F(\bullet)]_{p_2}\}(t_1)) && (1)\\
=& (\{x \mapsto t|_p\} \diamond \{F \mapsto t_1|_{p_1}[F(\bullet)]_{p_2}\})(\{x \mapsto t|_p\}(t_1)) && (2)\\
=& \{F \mapsto \{x \mapsto t|_p\}(t_1|_{p_1}[F(\bullet)]_{p_2})\}(\{x \mapsto t|_p\}(t_1)) && (3)\\
=& \{F \mapsto \{x \mapsto t_1|_{\hat{p}}\}(t_1|_{p_1}[F(\bullet)]_{p_2})\}(\{x \mapsto t_1|_{\hat{p}}\}(t_1)) && (4)\\
=& \{F \mapsto \{x \mapsto t_1|_{\hat{p}}\}(t_1)|_{p_1}[F(\bullet)]_{p_2}\}(\{x \mapsto t_1|_{\hat{p}}\}(t_1)) && (5)
\end{aligned}
$$

where (1) follows from definition of $\mathrm{R_{FF}}$ and $\mathrm{R_{x\neg F}}$, (2) follows from Lemma 5.3, which can be applied because $\mathsf{Vars}(\{x \mapsto t|_p\}) \cap \mathsf{Dom}(\{F \mapsto t_1|_{p_1}[F(\bullet)]_{p_2}\}) = \emptyset$ holds since $F \notin \mathsf{Vars}(t|_p)$ and $F \neq x$, (3) follows from Definition 5.1, in (4) the implicit definition of $\hat{p}$ holding $t_1|_{\hat{p}} = t|_p$ follows from Lemma 5.5, which can be applied since $t_1 \to_{\mathrm{R_{FF},p_1,p_2}} t$ and $F \notin \mathsf{Vars}(t|_p)$, and (5) holds because replacements of first-order variables and computation of subterms can be commuted in this way.

Finally, let the t' of the lemma be defined as $\{x \mapsto t_1|_{\hat{p}}\}(t_1)$ and note that $t_2 = \{F \mapsto t'|_{p_1}[F(\bullet)]_{p_2}\}(t')$. It remains to prove that t' can be derived from t_1, and that t_2 can be derived from t', as stated in the lemma. First, note that $t_1 \to_{\mathrm{R_{x\neg F},x,\hat{p}}} t'$ holds because neither F nor x occur in $t_1|_{\hat{p}}$ since $t_1|_{\hat{p}} = t|_p$ and, moreover, $x \in \mathsf{Vars}(t_1)$ holds because $x \in \mathsf{Vars}(t)$ and $\mathsf{Vars}(t) = \mathsf{Vars}(t_1)$. Now, note that $t' \to_{\mathrm{R_{FF},p_1,p_2}} t_2$ holds because:

- $F \in \mathsf{Vars}(t')$, since $F \in \mathsf{Vars}(t_1) \setminus \{x\} = \mathsf{Vars}(\{x \mapsto t_1|_{\hat{p}}\}(t_1)) = \mathsf{Vars}(t')$, and
- $F \notin \mathsf{Vars}(t'|_{p_1})$ since $F \notin \mathsf{Vars}(t_1|_{p_1}) \cup \mathsf{Vars}(\{x \mapsto t_1|_{\hat{p}}\})$ and thus $F \notin \mathsf{Vars}(\{x \mapsto t_1|_{\hat{p}}\}(t_1|_{p_1})) = \mathsf{Vars}(\{x \mapsto t_1|_{\hat{p}}\}(t_1)|_{p_1}) = \mathsf{Vars}(t'|_{p_1})$.

Hence, $t_1 \to_{\mathrm{R_{x\neg F},x,\hat{p}}} t' \to_{\mathrm{R_{FF},p_1,p_2}} t_2$ holds, which concludes the proof. ◀

▶ **Lemma 5.9.** *Let $\mathcal{F}$ be a ranked alphabet and let $\mathcal{V}$ be a set with first-order variables and a context variable F. Let t_1, t_2, t be terms in $\mathcal{T}(\mathcal{F}, \mathcal{V})$ such that $t_1 \to_{\mathrm{R_{xF},x_1,p_1}} t \to_{\mathrm{R_{x\neg F},x_2,p_2}} t_2$. Then, there exists a position $\hat{p}_2 \in \mathsf{Pos}(t_1)$ and a term $t' \in \mathcal{T}(\mathcal{F}, \mathcal{V})$ such that $t_1 \to_{\mathrm{R_{x\neg F},x_2,\hat{p}_2}} t' \to_{\mathrm{R_{xF},x_1,p_1}} t_2$ holds.*

The following result follows from the three previous lemmas, summarizing the goal of this section.

▶ **Lemma 5.10.** *Let $\langle \Delta, \mathcal{X}, F \rangle$ be a one-context unification instance. Let t be a term such that $\mathsf{term}(\Delta) \to^*_{\mathfrak{R}} t$ in n derivation steps. Then, there exists a derivation of length n of the form $\mathsf{term}(\Delta) \to^*_{\mathrm{R_{x\neg F}}} \to^*_{\mathrm{R_{FF}}} \to^*_{\mathrm{R_{xF}}} \to^{0,1}_{\mathrm{R_{FC}}} \to^*_{\mathrm{R_{x\neg F}}} t$.*

6 Complexity Analysis

We now have all the ingredients needed to prove that the one-context unification where the input terms are compressed using STG's is in NP. To prove this fact, we show, for each unifiable instance, that there exists a witness of polynomial size verifiable in polynomial time. In our setting, this witness is an STG generating the unified term and the verification consists of checking whether such a term is of a certain form.

▶ **Theorem 6.1.** *One-context unification with STG's is in NP.*

Proof. Let $\langle \Delta, \mathcal{X}, F \rangle$ be a one-context unification instance represented by an STG G and a nonterminal T of G such that $w_{G,T} = \mathsf{term}(\Delta)$. By Lemma 4.2, Δ has a solution if and only if there exists a derivation $\mathsf{term}(\Delta) \to^*_{\mathfrak{R}} t$ of length at most $2|\mathcal{X}| + 1$ such that t is of the form $\mathfrak{d}(\mathfrak{e}(u_1, u_1), \ldots, \mathfrak{e}(u_{|\Delta|}, u_{|\Delta|}))$. Moreover, by Lemma 5.10, we assume without loss of generality

that this derivation is of the form $\mathsf{term}(\Delta) \to^*_{\mathrm{R_{x\neg F}}} t_1 \to^*_{\mathrm{R_{FF}}} t_2 \to^*_{\mathrm{R_{xF}}} t_3 \to^{0,1}_{\mathrm{R_{FC}}} t_4 \to^*_{\mathrm{R_{x\neg F}}} t$, for some terms t_1, t_2, t_3, t_4.

We first prove that there exists an $\mathcal{F}$-extension G' of G such that $w_{G',T} = t$ and whose size is polynomially bounded by $|G|$. By Lemma 3.2, there exists an $\mathcal{F}$-extension G_1 of G such that $w_{G_1,T} = t_1$ and whose size is polynomially bounded by $|G|$. Now we claim that there exists an $\mathcal{F}$-extension G_2 of G_1 such that $w_{G_2,T} = t_2$ and whose size is polynomially bounded by $|G_1|$. By the conditions on the application of $\mathrm{R_{FF}}$ and by Lemma 5.5, the subcontexts computed in each step of the subderivation $t_1 \to^*_{\mathrm{R_{FF}}} t_2$ can be obtained from t_1. Hence, the sequence of applications of rules $\mathrm{R_{FF}}$ can be seen as a single application of a substitution of the form $\{F \mapsto c_1 \dots c_n F(\bullet)\}$, where each c_i is a subcontext of t_1. By Lemma 3.3, there exists an $\mathcal{F}$-extension G'_1 of G_1 such that $w_{G'_1,T} = t_1$ with a context nonterminal C such that $w_{G'_1,C} = c_1 \dots c_n$ and whose size is polynomially bounded by $|G_1|$ and $|\mathcal{X}|$. The STG G_2 mentioned above is defined as the $\mathcal{F}$-extension of G'_1 obtained by transforming the context variable into a context nonterminal generating $c_1 \dots c_n F(\bullet)$, where F is a freshly introduced terminal symbol standing for the context variable. Again by Lemma 3.2, there exists an $\mathcal{F}$-extension G_3 of G_2 such that $w_{G_3,T} = t_3$ and whose size is polynomially bounded by $|G_2|$. At this point, note that the exponent e involved in the application of $\mathrm{R_{FC}}$ can be at most exponential with respect to $|G_3|$ since it is linear with $|t_3|$. Hence, by Lemma 3.5, there exists an $\mathcal{F}$-extension G_4 of G_3 such that $w_{G_4,T} = t_4$ and whose size is polynomially bounded by $|G_3|$. Hence, by definition of G_1, G_2, G_3, G_4, it follows that $|G_4|$ is polynomially bounded by $|G|$. Finally, the existence of the grammar G' mentioned above follows from Lemma 3.2 and the fact that $|G_4|$ is polynomially bounded by $|G|$. Note that, as commented in Section 2.2, the size of the representation of G' is bounded by $|G'| \cdot (2 + m)$, where m is the maximum arity of the terminals of G'.

To conclude the proof, note that the property whether the term generated by a certain nonterminal, in our case by the nonterminal T of G', is of the form $\mathfrak{d}(\mathfrak{e}(u_1, u_1), \dots, \mathfrak{e}(u_{|\Delta|}, u_{|\Delta|}))$ can be checked in polynomial time with respect to the size of the given grammar. To see this, first note that checking whether the symbol labeling $w_{G',T}$ at position p, with $|p| = 0$ or $|p| = 1$, is $\mathfrak{d}$ or $\mathfrak{e}$, respectively, can be computed in linear time. Finally, nonterminals generating $w_{G',T}|_{i.1}$ and $w_{G',T}|_{i.2}$, for $i \in \{1, \dots, |\Delta|\}$, can be computed efficiently, and thus, by Lemma 2.2, checking whether $w_{G',T}|_{i.1} = w_{G',T}|_{i.2}$ can be solved in polynomial time. ◀

7 Conclusions and Further Work

We have proved that the one-context unification problem belongs to NP even when the input is compressed using STG's. A natural next step is to study whether there exists a polynomial time algorithm for this problem. However, this is also open when using an uncompressed term representation. Hence, it seems reasonable to first consider this problem before tackling the compressed case.

Another option is to study the complexity of particular cases of one-context unification with STG's. Concretely, the following one is particularly interesting due to its applications in term rewriting and querying XML-databases: solving an equation $F(s) \doteq t$ where t is ground and s does not contain the context variable F but may contain first-order variables. As shown in [19], this problem can be solved in polynomial time when the representation of s is noncompressing. However, it is still open whether the general case is NP-hard or it can be solved in polynomial time.

References

1 F. Baader and W. Snyder. Unification theory. In *Handbook of Automated Reasoning*, pages 445–532. Elsevier and MIT Press, 2001.

2 F. Bry and S. Schaffert. Towards a declarative query and transformation language for xml and semistructured data: Simulation unification. In *ICLP*, pages 255–270, 2002.

3 G. Busatto, M. Lohrey, and S. Maneth. Efficient memory representation of XML document trees. *Information Systems*, 33(4-5):456–474, 2008.

4 K. Erk and J. Niehren. Dominance constraints in stratified context unification. *Information Processing Letters*, 101(4):141–147, 2007.

5 A. Gascón, G. Godoy, and M. Schmidt-Schauß. Unification and matching on compressed terms. *ACM Transactions on Computational Logic*, 12(4):26, 2011.

6 A. Gascón, G. Godoy, M. Schmidt-Schauß, and A. Tiwari. Context unification with one context variable. *Journal of Symbolic Computation*, 45(2):173–193, 2010.

7 A. Gascón, S. Maneth, and L. Ramos. First-order unification on compressed terms. In *RTA*, pages 51–60, 2011.

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

9 S. Gulwani and A. Tiwari. Computing procedure summaries for interprocedural analysis. In *ESOP*, pages 253–267, 2007.

10 J. Levy, J. Niehren, and M. Villaret. Well-nested context unification. In *CADE*, pages 149–163, 2005.

11 J. Levy, M. Schmidt-Schauß, and M. Villaret. The complexity of monadic second-order unification. *SIAM Journal on Computing*, 38(3):1113–1140, 2008.

12 J. Levy, M. Schmidt-Schauß, and M. Villaret. On the complexity of bounded second-order unification and stratified context unification. *Logic Journal of the IGPL*, 19(6):763–789, 2011.

13 Y. Lifshits. Processing compressed texts: A tractability border. In *CPM*, pages 228–240, 2007.

14 M. Lohrey and S. Maneth. The complexity of tree automata and XPath on grammar-compressed trees. *Theoretical Computer Science*, 363(2):196–210, 2006.

15 M. Lohrey, S. Maneth, and R. Mennicke. Tree structure compression with RePair. In *DCC*, pages 353–362, 2011.

16 M. Lohrey, S. Maneth, and M. Schmidt-Schauß. Parameter reduction in grammar-compressed trees. In *FOSSACS*, pages 212–226, 2009.

17 W. Plandowski. Testing equivalence of morphisms on context-free languages. In *ESA*, pages 460–470, 1994.

18 K. Sadakane and G. Navarro. Fully-functional succinct trees. In *SODA*, pages 134–149, 2010.

19 M. Schmidt-Schauß. Pattern matching of compressed terms and contexts and polynomial rewriting. Frank report 43, Institut für Informatik. Goethe-Universität Frankfurt am Main, February 2011.