Polynomial Equality Testing for Terms with Shared Substructures

Manfred Schmidt-Schauß

Fachbereich Informatik, J.-W.-Goethe-Universität, Postfach 11 19 32,
D-60054 Frankfurt,Germany
Tel: (+49)69-798-28597, Fax: (+49)69-798-28919
schauss@ki.informatik.uni-frankfurt.de

Technical Report Frank-21
Research Group for Artificial Intelligence and Software Technology,
Institut für Informatik,
J.W.Goethe-Universität Frankfurt,

16.11.2005

Abstract. Sharing of substructures like subterms and subcontexts in terms is a common method for space-efficient representation of terms, which allows for example to represent exponentially large terms in polynomial space, or to represent terms with iterated substructures in a compact form. We present singleton tree grammars as a general formalism for the treatment of sharing in terms. Singleton tree grammars (STG) are recursion-free context-free tree grammars without alternatives for non-terminals and at most unary second-order nonterminals. STGs generalize Plandowski's singleton context free grammars to terms (trees). We show that the test, whether two different nonterminals in an STG generate the same term can be done in polynomial time, which implies that the equality test of terms with shared terms and contexts, where composition of contexts is permitted, can be done in polynomial time in the size of the representation. This will allow polynomial-time algorithms for terms exploiting sharing. We hope that this technique will lead to improved upper complexity bounds for variants of second order unification algorithms, in particular for variants of context unification and bounded second order unification.

Keywords: Sharing – tree grammars – polynomial word problem

1 Introduction

Sharing of substructures is a classic and ubiquitous method for efficient space usage in software systems. In Automated Deduction, sharing of subterms and subcontexts in terms is a common method for space-efficient representation of terms, which allows for example to represent exponentially large terms in polynomial space, or to represent terms with iterated substructures in a compact form. An example is first-order unification [Rob65,MM82], which is known to be polynomial, and where terms in a most general unifier may be exponentially large, but can be represented in linear space using sharing of subterms. In Automated Deduction Systems, methods like Coded Context Trees are used for compression and indexing of terms, which exploit term sharing and context sharing, however, without the possibility of composing contexts; see [GNN04].

A first formalization of reasoning about sharing of words was done using a specialized form of grammars: [Pla94] introduced a grammar formalism, now called singleton context free grammars (SCFG), which is a very convenient and rather general formalism for representing large words– using only sharing – in a compact form, and also for reasoning about complexity of algorithms on words with shared subwords. The grammars are non-recursive, have exactly one rule per non-terminal, and are Chomsky grammars, i.e. the right hand side of rules has at most length 2. Plandowski has proved that in an SCFG, the words w_A, w_B represented by two nonterminals A, B, can be compared for equality in time polynomial in the size of the grammar. Usage of these grammars is called grammer compression (also straight-line program) in literature on word-compressing algorithms (see [Ryt04]). For example, $((ab)^{100}c^{20})^{1000}$ is a specialized representation of a word of length 220000, which can easily be represented (using sharing) by a nonterminal in an SCFG of polynomial size in the size of $((ab)^{100}c^{20})^{1000}$, i.e. 16. For example, Plandowski's theorem, resp. algorithm, can be applied to efficiently show that $((ab)^{300}a(ba)^{100}b)^{200} = (ab)^{80200}$, after an easy encoding.

Terms, also called ranked trees, generalize words. The notions of grammars and automata can also be generalized to terms (see e.g. [CDG$^+$97]). In this paper we define singleton tree grammars (STGs) as a generalization of SCFG in order to have a general mechanism for a compressed representation of terms using sharing. We will employ the restriction that the arity of nonterminals is 0 or 1, i.e. there are tree nonterminals, and context-nonterminals representing trees with a single hole. STGs are recursion-free context-free tree grammars without alternatives for non-terminals and at most unary second-order nonterminals. The right hand sides of rules are flat (second-order) terms, i.e the size is at most $maxarity + 1$, where $maxarity$ is the maximal arity of a function symbol occurring in the grammar. Seen from a tree grammar view, we assume that there is only one bound variable in the term functions, and that the right hand sides of rules contain at most one occurrence of a bound variable, i.e., they are linear in the lambda-bound variable. In our formalism of STGs, these lambda-bound variable are not mentioned explicitly. The terms represented by an STG may be exponentially large and also exponentially deep.

The goal of this paper is to generalize Plandowski's theorem to singleton tree grammars (see Theorem 9.13), which states that the word problem for two tree nonterminals in an STG is solvable in polynomial time. The proof is given by generalizing Plandowski's algorithm to terms. This generalized algorithm uses expansion and compaction, which can also be used for STGs. The structure of Plandowski's proof and several of Plandowski's lemmas can be transferred to STGs, however, there are far more cases and also new notions are to be introduced, e.g., a periodicity lemma for overlaps of a word w with its single-point mutants has to be proved.

A planned future application of the theorem will be sharper upper complexity bounds for variants of second-order such as bounded second-order unification [SS04] and algorithms for variants of context unification [SS02,SS01]. We conjecture that the results are also extendible to contexts with several holes. It is unclear whether the result can be extended if nonlinear lambda terms are allowed in singleton tree grammars. We leave both issues for future work.

There are already methods to represent (sets of) terms in compact form using exponents: ρ-terms [CH95], I-terms [Com95], R-terms [Sal92], and primal grammars [HG97]. However, usually these representations are intended to represent infinite sets of terms and to support unification algorithms. Usually, these formalisms allow integer variables in terms instead of integers, which is more general than STGs, however, I-terms and R-terms do not allow nesting of exponents. The considered word problem is usually different from ours (see [HS98]). Our result can be applied to comparing I-terms for equality if the integer exponents are already instantiated, and may be applied in an estimation of the complexity of unification of I-terms. The result cannot be used for R-terms, since this would require that nonlinear lambda terms are permitted in the tree grammar formalism.

The paper is structured as follows. After recalling some basic definitions and singleton context free tree grammars and their properties, we introduce the singleton tree grammars in section 4 and explore some of their required properties in section 5. In section 6 we define sets of overlap-triples as data structure and in section 7 we define the for the algorithm *WOPS* solving the word problem efficiently. In sections 8 and 9 it is shown that the algorithm correctly solves the word problem and that it has the claimed polynomial complexity.

2 Preliminaries

Let Σ be a finite signature of function symbols. Every function symbol comes with an arity, denoted $ar(f)$, which is a nonnegative integer. The number *maxarity* is defined to be the maximal arity of function symbols in the signature Σ. Function symbols with $ar(f) = 0$ are also called *constant symbols.* We assume that the signature contains at least one constant symbol.
First-order terms (also called ranked trees) t are formed using the grammar rule $t ::= f(t_1, \ldots, t_{ar(f)})$ where f is a function symbol of arity n, and t_i for $i = 1, \ldots, n$ are terms. For a constant symbol a, we write a instead of $a()$. Terms are denoted using lower case letters s, t. We call f the *head* of the term $t = f(t_1, \ldots, t_{ar(f)})$, denoted as $head(t)$. We will use words as tree addresses for pointing to subterms in a term, and call them *positions.* With $Pos(t)$ we denote the set of positions of t. For $p \in Pos(t)$, the subterm of t at position p is denoted as $t_{|p}$. The notation $t[s]_p$ means that t has a subterm s at position p. For words

w_1, w_2 we write $w_1 \perp w_2$, iff neither w_1 is a prefix of w_2 nor w_2 is a prefix of w_1. We write $w_1 \parallel w_2$, otherwise, i.e. iff w_1 is a prefix of w_2 or w_2 is a prefix of w_1.

A *context* W is a first order term over the signature extended with a 0-ary constant, denoted $[\cdot]$, also called the hole, such that there is exactly one occurrence of the hole in W. Usually, contexts are denoted as $C[.]$, or also simply as C. The term that results from plugging in the term t in the hole of the context $C[.]$ is denoted as $C[t]$. Similarly, we use this notation for contexts. i.e. the context that results from $C_1[.]$ by plugging in the context $C_2[.]$ is denoted as $C_1[C_2[.]]$, or also as C_1C_2. The notation C^n means $\underbrace{C \dots C}_{n}$. Another view is that contexts are functions on terms, e.g. $C[.]$ is the function $\lambda x.C[x]$, where we will always have the restriction that the body of the abstraction has exactly one occurrence of the bound variable x, i.e., is a linear term in x. With $mp(w_C)$ we denote the position of the hole in the context $C[.]$; this position is also called the *main path* of C. The number $|mpC|$ is also called the *main depth* of C. With $Pos(C)$ we denote the set of positions of C, excluding the position of the hole. If $p \in Pos(C)$, the subcontext of C at position p is denoted as $C_{|p}$. In the case that $mp(w_C)$ is a prefix of p, we also permit the notation $C_{|p}$ and define $C_{|p} := [\cdot]$.

A context W can be seen as a word of length $|mp(W)|$ by representing it as a concatenation of $|mp(w_C)|$ contexts of main depth 1. So we can use definitions, tools and properties for words also for contexts. A context W has a *period* v, if W, seen as a word, has the period v, i.e., there is a context W' with $|mp(W')| = v$, and W is a subword of $(W')^n$, for some appropriate positive integer n. This definitions makes only sense, if $v < |mp(W)|$.

From [Lot83] we see that the following holds:

Lemma 2.1. *If a context W has two periods v_1, v_2 with $v_1 + v_2 \leq |mp(W)|$, then W has also a period* $\gcd(v_1, v_2)$.

Given a signature Σ of functions symbols and a set $\Sigma^{(2)}$ of unary second-order symbols, we define also second-order terms as follows: $t ::= f(t_1, \dots, t_n) \mid X(t)$, where f is an n-ary function symbol from Σ, and X a second-order symbol in $\Sigma^{(2)}$. Using the hole $[.]$ as a constant, we can form second order terms containing this hole. Throughout this paper, we allow only one occurrence of $[.]$ in a second order term. Second order terms containing a single constant $[.]$ are also called *second order context terms.*

If we speak of "terms" in the following, we always mean first order terms; if we mean second order terms, this is mentioned explicitly.

3 Singleton Context Free Grammars

We repeat the definitions and results of Plandowski [Pla94].

3.1 Singleton Context Free Grammars and Plandowski's Theorem

Definition 3.1. *A* singleton context free grammar (SCFG) *is a CFG $G = (\mathcal{T}, \mathcal{N}, R)$ where $\mathcal{T}$ are the terminals, $\mathcal{N}$ are the nonterminals with $\mathcal{N} \cap \mathcal{T} = \emptyset$,* *and $R \subseteq \mathcal{N} \times (\mathcal{T} \cup \mathcal{N})^*$ are the rules, such that the right hand sides of R are words of length ≤ 2. For every nonterminal N, there is exactly one rule in R of the form $N \to t$. The grammar must be non-recursive. More formally, the relation $\xrightarrow{+}$ has no cycles, where $\xrightarrow{+}$ is the transitive closure of the relation $\to \ \subseteq \mathcal{N} \times \mathcal{N}$, where $N \to M$ iff there is a rule $N \to t \in R$ where M occurs in t.*

The word generated by a non-terminal N is denoted as $w_{G,N} \in \mathcal{T}^*$. If the grammar is clear from the context, we drop the suffix G. We extend the notation w_N to arbitrary words $\alpha \in (\mathcal{T} \cup \mathcal{N})^*$ such that $w_\varepsilon = \varepsilon$, $w_a = a$ for terminals a, and $w_{a_1 \ldots a_n} = w_{a_1} \ldots w_{a_n}$.

Note that the start-symbol is omitted in the definition, since we are interested in the words generated by each non-terminal.

Theorem 3.2. *Given an SCFG G, and two nonterminals N_1, N_2. Then it is decidable in polynomial time in the size of G, whether $w_{N_1} = w_{N_2}$.*

Proof. In [Pla94].

An equivalent formulation is to have two different grammars G_1, G_2 and a nonterminal N_i from G_i for $i = 1, 2$, and asking whether N_1, N_2 define the same word.

Example 3.3. Given a language for representing words with the syntax
$P ::= a \mid b \mid P^n \mid (P) \mid PP$
where a, b are terminals, and n is a positive integer, it is possible to write a^{1000} to represent a word $\underbrace{a \ldots a}_{1000}$. More involved examples are $ab^{100}(ab^{100}a)^{15}$. Testing equality of the represented words takes exponential time if the comparison is done by first expanding them. A nontrivial test which should succeed is for example whether $(ab)^{100}a = a(ba)^{100}$.
It is easy to encode this equality test in a small SCFG, such that the number of rules is polynomial in the printed size of this equation (which is 21 for this equation). E.g.: a^{1000} can be encoded along its representation as binary number 1111101000 using the rules $N_1 ::= a, N_2 ::= N_1N_1, N_3 ::= N_2N_2, \ldots, N_9 ::= N_8N_8, M_9 ::= N_9M_8, M_8 ::= N_8M_7, M_7 ::= N_7M_6, M_6 ::= N_6M_5, M_5 ::= N_5N_3$. The nonterminal M_5 encodes a^{1000}. After the appropriate encoding, Theorem 3.2 assures us that equality tests are possible in polynomial time.

Definition 3.4. *The size of an SCFG G is the number of its rules and denoted as $|G|$.*
The depth(D) of a nonterminal D is defined as:
$depth(D) := 1 + \max(depth(D_1), depth(D_2))$ if the rule for D is $D = D_1D_2$.
The depth of G is the maximum of the depths of all its nonterminals.

3.2 Operations on Singleton Context Free Grammars

In this section we define several operations on an SCFG G, which may be tests for specific properties, or extending the grammar, and we also explore the complexity of these operations.

Definition 3.5. *An* extension *of an SCFG* $G = (\mathcal{T}, \mathcal{N}, R)$ *is a SCFG* $G' = (\mathcal{T}', \mathcal{N}', R')$ *with* $\mathcal{T} \subseteq \mathcal{T}', \mathcal{N} \subseteq \mathcal{N}'$*, and* $R \subseteq R'$*.*

An extension *of an STG* $(\mathcal{TN}, \mathcal{CN}, \Sigma, R)$ *is a STG* $G' = (\mathcal{TN}', \mathcal{CN}', \Sigma', R')$ *with* $\mathcal{TN} \subseteq \mathcal{TN}'$*,* $\mathcal{CN} \subseteq \mathcal{CN}'$*,* $\Sigma \subseteq \Sigma'$*, and* $R \subseteq R'$*.*

Extending the grammar during the run of an algorithm and estimating the size increase has to be done carefully, since a polynomial size increase in every step is in general too large.

Lemma 3.6. *Given an SCFG* G*, a nonterminal* N*, and a number* $0 < n < |w_N|$*, an extension* G' *of* G *can be constructed in polynomial time in* $|G|$*, such that* G' *contains a nonterminal* N' *such that* $|N'| = n$*, and* $w_{N'}$ *is a suffix (or a prefix, respectively) of* w_N*. Moreover,* $|G'| \leq |G| + \mathit{depth}(G)$*,* $\mathit{depth}(G') = \mathit{depth}(G)$*.*

Proof. See [LSSV04].

Lemma 3.7. *Given an SCFG* G*, a nonterminal* N*, and a number* $n > 0$*, an extension* G' *of* G *can be constructed in polynomial time in* $|G|$ *and* $\log(n)$*, such that* G' *contains a nonterminal* N' *and* $w_{N'} = {w_N}^n$*. Moreover,* $|G'| \leq |G| + 2 * \lceil \log(n) \rceil$*,* $\mathit{depth}(G') \leq \mathit{depth}(G) + \lceil \log(n) \rceil$*.*

Proof. See [LSSV04].

Lemma 3.8. *Given an SCFG* G*, nonterminals* $N_1, \ldots, N_n$*, an extension* G' *of* G *can be constructed in polynomial time in* $|G|$ *and* n*, such that* G' *contains a nonterminal* N' *such that* $w_{N'} = w_{N_1} \ldots w_{N_n}$*. Moreover,* $|G'| \leq |G| + n - 1$*,* $\mathit{depth}(G') = \mathit{depth}(G) + \lceil \log(n) \rceil$*.*

Proof. See [LSSV04].

Lemma 3.9. *Given an SCFG* G *and two nonterminals* N_1, N_2*, the following tests are polynomial in* $|G|$*:*

- w_{N_1} *is a prefix of* w_{N_2}*.*
- $w_{N_1} \perp w_{N_2}$*.*
- $w_{N_1} \parallel w_{N_2}$*.*

Proof. It is sufficient to prove the first claim. The other claims can be derived immediately. As already proved in [Pla94], the computation of $|w_a|$ can be done in polynomial time. Given N_1, N_2, we compute the lengths $|w_{N_1}|$ and $|w_{N_2}|$. If $|w_{N_1}| > |w_{N_2}|$, then $|w_{N_1}|$ cannot be a prefix of $|w_{N_2}|$. Otherwise, construct an extension of the grammar G that contains a nonterminal N_3 such that w_{N_3} is a prefix of w_{N_2} and $|w_{N_3}| = |w_{N_1}|$. This is possible in polynomial time due to Lemma 3.6, and the size of the new STG is at most polynomial in the size of G. Then we compare w_{N_1} and w_{N_3}, which can also be done in polynomial time due to Theorem 3.2.

4 Singleton Tree Grammars

We define singleton tree grammars as a generalization of singleton CFGs. We consider languages of ranked trees and extend the expressivity of SCFGs by permitting contexts. The definition is consistent with the context free tree grammars in [CDG+97], however a special case.

Definition 4.1. *A* singleton tree grammar (STG) *is a tree grammar, i.e. a 4-tuple $(\mathcal{TN}, \mathcal{CN}, \Sigma, R)$, where $\mathcal{TN}$ are tree nonterminals, $\mathcal{CN}$ are context non-terminals, and Σ is a signature of function symbols (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:*

- *$A ::= f(A_1, \dots, A_n)$, where A, A_i are tree non-terminals, and $f \in \Sigma$ is an n-ary function symbol.*
- *$A_1 ::= C[A_2]$ where A_1, A_2 are tree nonterminals and C is a context nonterminal.*
- *$C_1 ::= C_2C_3$, where C_i are context non-terminals.*
- *$C ::= f(A_1, \dots, A_{i-1}, [\cdot], A_{i+1}, \dots, A_n)$, where A_i are tree nonterminals, C is a context non-terminal, $[\cdot]$ is the hole, and $f \in \Sigma$ an n-ary function symbol.*

The tree grammar must be non-recursive. More formally, the relation $\xrightarrow{+}$ has no cycles, where $\xrightarrow{+}$ is the transitive closure of the relation $\rightarrow \ \subseteq \mathcal{N} \times \mathcal{N}$, where $N \rightarrow M$ iff there is a rule $N \rightarrow t \in R$ where M occurs in t.

Furthermore, for every non-terminal N there is exactly one rule having N as left hand side.
The derivation using the rules is performed for second order terms over a signature $\Sigma \cup \mathcal{TN}$, where the tree nonterminals have zero arity, and a second order signature $\mathcal{CN}$. The derivation starts with A as second order term, where A is a tree nonterminal, or with the second order context term $C([.])$, where C is a context nonterminal. The rules are used in the derivation process as follows.

- *If $A ::= f(A_1, \dots, A_n)$ or $A ::= C[A']$, then A in a second order term or second order context term is replaced by $f(A_1, \dots, A_n)$, or $C(A')$, respectively.*
- *If $C_1 ::= C_2C_3$, then $C_1(s)$ in a second order term or or second order context term is replaced by $C_2(C_3(s))$.*
- *If $C ::= f(A_1, \dots, A_{i-1}, [\cdot], A_{i+1}, \dots, A_n)$, then $C(s)$ in a term or context is replaced by $f(A_1, \dots, A_{i-1}, s, A_{i+1}, \dots, A_n)$.*

The derivation process stops if there are no more replacement possibilities, and the result is either a first order term or a context. We write $\alpha \rightarrow_G \beta$ if there is a derivation starting with the second order term α and reaching the second order term β. Given a non-terminal A, or C, respectively, of G, by $w_{G,A}$ or $w_{G,C}$ we denote the generated first order term or first order context, respectively. More general, if t is a second order term or a second order context term over the signature given above, then $w_{G,t}$ is the generated first order term or the context, respectively. If the grammar G is clear, we omit the suffix in our notation. We also may use the notation $mp(w_C)$ as a short hand for $mp(w_C)$.

Note that we omit the rules of the form $A ::= A', C ::= C', C ::= [\cdot]$ in the definition, since these rules can easily be eliminated (see Remark 4.4).

Definition 4.2. *The size of a grammar (STG) G is the number of its rules and denoted as $|G|$.*
The depth(D) of a nonterminal D is defined as:

- *$depth(D) := 1 + \max(depth(D_1), depth(D_2))$ if the rule for D is $D = D_1D_2$ or $D_1[D_2]$.*
- *$depth(D) := 1 + \max\{depth(A_j) \mid j = 1, \ldots, n\}$ if the rule for D is $D = f(A_1, \ldots, A_n)$, where we presume that the depth of the hole $[\cdot]$ is zero and the maximum over an empty set is 0.*

The depth of a grammar is the maximum of the depths of all nonterminals.
In an STG G, we write $A > B$, iff one of the following holds: $A \in \mathcal{TN}$, $A \xrightarrow{+} \alpha$ for a second-order term α, and B occurs in α; or $A \in \mathcal{CN}$, $A([.]) \xrightarrow{+} \alpha$ for a second-order context term α, and B occurs in α.

Example 4.3. If C is the context $C = f(g(x), [\cdot], h(x))$, then the term $C^{100}(b)$ can be represented using a singleton tree grammar as follows:

$$\begin{array}{lll}
 & C_1 ::= C_0C_0 & \\
C_0 ::= f(A_1, [\cdot], A_2) & C_2 ::= C_1C_1 & D_1 ::= C_6C_5 \\
A_1 ::= g(A_3) & C_3 ::= C_2C_2 & D_2 ::= D_1C_2 \\
A_2 ::= h(A_3) & C_4 ::= C_3C_3 & B ::= b \\
A_3 ::= x & C_5 ::= C_4C_4 & D_3 ::= D_2[B] \\
 & C_6 ::= C_5C_5 &
\end{array}$$

The nonterminal D_3 represents the term

$$f(g(x), f(g(x), f(\ldots(b)\ldots), h(x)), h(x)),$$

where the nesting depth is 100. In an exponent notation for contexts as unary functions with $C \equiv \lambda z.f(g(x), z, h(x))$, it could be represented as $C^{100}(b)$.

The goal is to prove the following theorem (see Theorem 9.13).

Theorem 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 proof will be given in the following sections.
An equivalent problem is the question, given two different STGs G_1, G_2 and N_i from G_i for $i = 1, 2$, whether $w_{N_1} = w_{N_2}$.

Remark 4.4. Note that it would be possible to permit more kinds of rules in an STG. Suppose we also allow rules of the form:

- $C_1 ::= C_2$.
- $A_1 ::= A_2$.
- $C_1 ::= [\cdot]$.

These rule formats allow a more compact representation of the terms in certain cases.

The first two kinds of rules can be eliminated by replacing e.g. the rule $C_1 ::= C_2$ by the rule $C_1 ::= W$, where $C_2 ::= W$ is the rule for C_2. Similarly for the second rule. The complexity will slightly increase, since there may be a size increase of $O(|G|^2)$. The third rule is irrelevant for the worst case complexity of the word problem.

5 Operations on Singleton Tree Grammars Using SCFGs for Positions

Lemma 5.1. *Given an STG G and a nonterminal D from G. Then $head(w_D)$ can be computed in polynomial time in $|G|$.*

Proof. The grammar is nonrecursive. Thus the number of steps is at most $depth(G)$, since in the right hand sides of rules only $C[A]$ and CC' are non-trivial, and only C needs to be further expanded.

Corollary 5.2. *Given an STG G and two nonterminals D_1, D_2 from G. Then the test $head(w_{D_1}) = head(w_{D_2})$ can be performed in polynomial time in $|G|$.*

Definition 5.3. *Given an STG G, an SCFG G' with positive integers as terminals, and a (partial) injective mapping $\mathcal{CN}_G \to \mathcal{N}_{G'}$, such that $C \mapsto C_s$ for every context nonterminal. If $mp(w_{G,C}) = w_{G',C_s}$ for every $C \in \mathcal{CN}_G$, then G' is called a* position grammar *of G.*

Definition 5.4. *Given an STG G. Then we define an SCFG G_{pG} with positive integers as terminals, as follows:*
For every context nonterminal C in G, there is a unique nonterminal C' in G_{pG}. The rules are as follows:

- *$C' ::= i$ if $C ::= f(A_1, \ldots, A_{i-1}, [\cdot], A_{i+1}, \ldots, A_n)$ is a rule in R_G.*
- *$C' ::= C'_1 C'_2$ if $C ::= C_1 C_2$ is a rule in R_G.*

Lemma 5.5. *Given an STG G, then G_{pG} can be constructed in polynomial time.*
Moreover, G_{pG} is a position grammar of G.

Proof. By induction.

Lemma 5.6. *Given an STG G and a position grammar G_s. For a nonterminal D from G and a nonterminal N from G_s the test whether $w_N \in Pos(w_D)$ can be performed in polynomial time in $|G|$ and $|G_s|$.*

Proof. We show that a recursive algorithm using smaller and smaller STGs G has at most $|G|$ recursions and every step is polynomial. Let $d = depth(G)$.

- Consider the case that D is a context nonterminal. Then the nonterminal N_D corresponding to D is already in G_s. The computation of a nonterminal N_1 representing the longest common prefix of w_N and w_{N_D}, a number k, and a nonterminal N_2 representing the suffix, such that $w_{N_1} k w_{N_2} = w_N$ can be done in polynomial time giving G'_s with $|G'_s| \leq |G_s| + 2 * d$. Determining the right hand side component in G, where the derivation starting from D forks for N and N_D, can be done in polynomial time and results in a component $f(A_1, \dots, A_{i-1}, [\cdot], A_{i+1}, \dots, A_n)$, and a number $k \neq i$. Then the computation can proceed recursively using a smaller grammar G' of all nonterminals necessary to define the tree nonterminal A_k, and the nonterminal N_2.
- Let D be a tree nonterminal. Then either $D ::= f(A_1, \dots, A_n)$, and we can argue as above. If $D ::= C[A]$, then again we can compare N with the nonterminal C_s for $mp(w_C)$. If $mp(w_C)$ is a prefix of w_N, then we construct a suffix nonterminal N_1 with $w_N = mp(w_C) w_{N_1}$, and use then the same algorithm for A and N_1. If $mp(w_C) \perp w_N$, we call the same procedure using C and N.

It is clear that the number of recursive calls is at most $|G|$, since in every call at least one rule can be removed from the current G. Hence the size of the final G''_s is at most $|G_s| + 2|G| * d \leq |G_s| + 2|G|^2$. Thus there is a size increase that is overall bounded by a polynomial, all intermediate steps are polynomial. We can estimate the sum of all steps and obtain a polynomial upper bound in $|G|$ and $|G_s|$. □

6 Triples for a Representation of Overlappings

We define overlay triples as a data structure for the algorithm solving the word problem of STGs.

Definition 6.1. *Two contexts or trees W_1, W_2 have a* top-overlap*, iff for every position $p \in Pos(W_1) \cap Pos(W_2)$: $head(W_1|_p) = head(W_2|_p)$, i.e. the function symbols at all common positions are the same, where the position of the hole and the positions below the hole of contexts are ignored.*

Definition 6.2. *Let G be a STG, and let G_s be a position grammar of G. A* triple *(wrt. G and G_s) is either of the form (A, B, ε) or of the form (A, B, N) where A, B are non-terminals from G, and N is a non-terminal in G_s.*
A triple (A, B, N) is well-formed*, iff $w_N \in Pos(w_B)$, or if w_B is a context and $mp(w_B)$ is a prefix of w_N. Triples (A, B, ε) are always well-formed.*
A well-formed triple (A, B, N) is valid*, iff $(w_B)_{|w_N}$ and w_A have a top-overlap.*

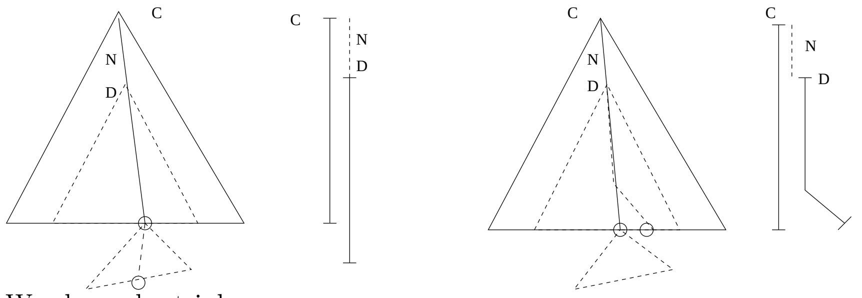
Definition 6.3. *Assume given an STG G and a position grammar G_s. The valid triples are partitioned into the following classes:*

- ε-triples*: triples of the form (D_1, D_2, ε).*

– inclusion triples*: Triples of the form (D_1, D_2, N), such that one of the following holds:*
  1. *D_2 is a tree nonterminal, or*
  2. *D_2 is a context non-terminal with $w_N \perp mp(w_{D_2})$.*
  3. *D_2 is a context non-terminal with $w_N \parallel mp(w_{D_2})$, and $w_N mp(w_{D_1})$ is a prefix of $mp(w_{D_2})$.*
– overlay triples*: (D, C, N), where C is a context nonterminal, $w_N \parallel mp(w_C)$, and one of the following holds:*
  1. *D is a tree nonterminal ((D, C, N) is a* tree overlay triple*), or*
  2. *D is a context nonterminal, and $w_N mp(w_D)$ is not a prefix of $mp(w_C)$. These are further distinguished into*
     - parallel overlay triples*, if $mp(w_C) || w_N mp(w_D)$.*
     - orthogonal overlay triples*, if $mp(w_C) \perp w_N mp(w_D)$.*

We say that the ε-triples and the inclusion triples are transient *triples, since the algorithm WOPS will try to eliminate them first. The nontransient triples are the overlay triples.*

We illustrate the parallel and orthogonal overlay triple (D, C, N). From left to right, the first two diagrams show a parallel overlay triple, a full tree picture and a picture showing only the main paths, whereas the third and fourth shows the same for an orthogonal



We also order triples:

Definition 6.4. *For $i = 1, \dots, 4$ let D_i be nonterminals in an STG G. Given two pairs (D_1, D_2) and (D_3, D_4), we say $(D_1, D_2) > (D_3, D_4)$, iff $D_1 \geq D_3$, $D_2 \geq D_4$ and one of the two relations is strict, where the base ordering is the derivation ordering w.r.t. G (see Definition 4.1).*
Given two triples (D_1, D_2, N) and (D_3, D_4, N'), we write $(D_1, D_2, N) > (D_3, D_4, N')$, iff $(D_1, D_2) > (D_3, D_4)$ or $(D_1, D_2) > (D_4, D_3)$.

Note that the ordering is exactly the multiset-ordering for $\{D_1, D_2\}$ and $\{D_3, D_4\}$ with base ordering $>$ on nonterminals.

7 The Algorithm *WOPS*

In this section we define the algorithm *WOPS* for solving an equation $N_1 = N_2$ of nonterminals of an STG using transformation rules. The data structure is mainly a set of triples. However, for control purposes, the triples are partitioned into different sets.

The main algorithmic parts are:

- straightforward unfolding of triples by transformations, however controlled by a strategy.
- Simplification rules for triples
- Failure Rules for compaction.
- Compaction rules for overlay triples

Definition 7.1. *The algorithm WOPS has as input an STG G and two tree nonterminals N_1, N_2. It has as state a tuple $(G_s, S_{\varepsilon,closed}, S_{\varepsilon,open}, S_{not}, S_{ot})$, consisting of a singleton context free grammar G_s as position grammar of G, a set $S_{\varepsilon,closed}$ of ε-triples already expanded, a set $S_{\varepsilon,open}$ of ε-triples not yet expanded, a set S_{not} containing the inclusion triples, and a set S_{ot} of overlay triples.*

1. *The* initial state *is $(G_{pG}, \emptyset, \{(N_1, N_2, \varepsilon)\}, \emptyset, \emptyset)$, where G_{pG} is the position grammar of G (see Definition 6.2), and N_1, N_2 are the tree nonterminals that have to be compared for equality.*
2. Halting *The algorithm WOPS returns "YES", i.e. it halts with success, if there are no more triples to be selected, i.e., if $S_{\varepsilon,open} = S_{ot} = S_{not} = \emptyset$. It halts with failure (i.e., returns "NO"), if an explicit Fail is the result of a simplification or transformation.*
3. *Apply the simplification rules in subsection 7.1 and the failure rules in subsection 7.3 exhaustively.*
4. *Apply the compaction rules in subsection 7.4 exhaustively.*
5. *Apply a transformation rule according to the following strategy:*
   A triple T is selected from one of the sets in the state according to the following priorities:
   - *S_{not},*
   - *$S_{\varepsilon,open}$*
   - *S_{ot}, where a maximal triple according to Definition 6.4 in S_{ot} has to be selected.*

   After the selection, the following is performed:
   If $T \in S_{not} \cup S_{ot}$, then T will be removed from each set.
   If $T \in S_{\varepsilon,open}$, then it will be removed from $S_{\varepsilon,open}$ and added to $S_{\varepsilon,closed}$.

   The selected triple T will then be subject to a transformation as defined below in subsection 7.2, which may extend the singleton grammar G_s to G'_s. If the triples $T_1, \ldots, T_n$ are the outcome of the transformation, then do the following for each triple T_i:
   - *If T_i is an ε-triple, and $T_i \in S_{\varepsilon,open} \cup S_{\varepsilon,closed}$, then do nothing, otherwise add it to the set $S_{\varepsilon,open}$.*
   - *Otherwise, if T_i is an overlay triple, then add it to S_{ot}.*
   - *Otherwise, if T_i is an inclusion triple, then add it to S_{not}.*
6. *For iterating this cycle, go to step 2*

7.1 The Transformation Rules of the Algorithm *WOPS*

First we define simplification rules on the state.

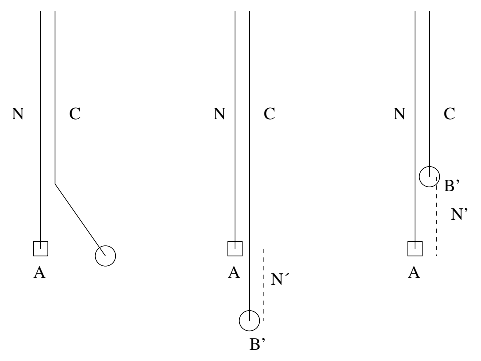
Simplification Rules The simplification rules operate on the state. The simplification rules are tested in the stated sequence, i.e. the simplification rule j' is applied only if the simplification rules $j < j'$ are not applicable.
We write D, D_i if we mean a tree or context nonterminal.

1. If there is some $(D_1, D_2, \varepsilon) \in S_{\varepsilon,open}$ with $head(w_{D_1}) \neq head(w_{D_2})$, then Fail: The initial equation is False.
2. Remove (D, C, N) from S_{ot} or S_{not}, if $mp(w_C)$ is a prefix of w_N.
3. If there is some $(D_1, D_2, N) \in S_{ot} \cup S_{not}$ such that $w_N \notin Pos(w_{D_2})$, then Fail: The initial equation is False.
4. If there is some $(D_1, D_2, N) \in S_{ot} \cup S_{not}$ with $head(w_{D_2}|_{w_N}) \neq head(w_{D_1})$, then Fail: The initial equation is False.
5. If there is some $(A, B, N) \in S_{ot} \cup S_{not}$, where w_A or w_B is a constant, then remove this triple from $S_{ot} \cup S_{not}$. If there is a triple $(A, B, \varepsilon) \in S_{\varepsilon,open}$, where w_A or w_B is a constant, then move it to $S_{\varepsilon,closed}$.

7.2 Transformation Rules

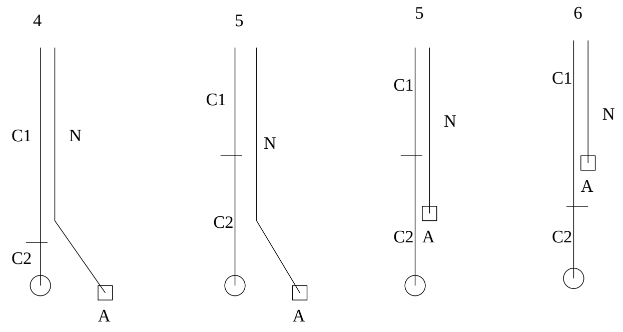
We assume that the simplification rules are applied exhaustively to the state before the transformation rules are applied.
Note that in the following rules, the notation (A, B, N) means that $w_N \neq \varepsilon$. The main case distinction are four cases, depending on the types of the nonterminals in the triples: (tree, tree), (tree, context), (context, context) and (context, tree). In some cases, the position grammar G_s is extended to a new position grammar G'_s, since a new nonterminal has to be constructed. In the following we use diagrams for illustrating the relation of the main path of contexts and a position nonterminal. Usually, this is sufficient to understand and analyse the possibilities, ignoring the rest of the context.

– (T,T)-case. An illustration of the cases 4 to 6 is:



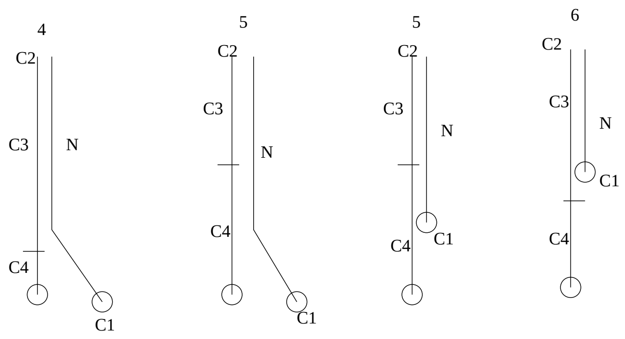
1. $(A, B, \varepsilon) \to \{(A_1, B, N_1), \dots, (A_k, B, N_k)\}$ if $A ::= f(A_1, \dots, A_k)$, where G'_s contains the nonterminals $N_1, \dots, N_k$ with rules $N_i ::= i$ for $i = 1, \dots, k$. Note that due to simplification, $ar(f) > 0$.
2. $(A, B, \varepsilon) \to \{(C, B, \varepsilon), (A', B, N)\}$ if $A ::= C[A']$, where N is the nonterminal in G_s with $w_N = mp(w_C)$.
3. $(A, B, N) \to (A, B_i, N')$ if $B ::= f(B_1, \dots, B_k)$, and $w_N = iw_{N'}$, where N' is a nonterminal in G'_s.
4. $(A, B, N) \to (A, C, N)$ if $B ::= C[B']$, and $w_N \perp mp(w_C)$.
5. $(A, B, N) \to \{(A, C, N), (B', A, N')\}$ if $B ::= C[B']$, and w_N is a proper prefix of $mp(w_C)$, where N' is a nonterminal in G'_s with $w_N w_{N'} = mp(w_C)$, otherwise.
6. $(A, B, N) \to (A, B', N')$ if $B ::= C[B']$, and $mp(w_C)$ is a prefix of w_N, where $N' = \varepsilon$ if $w_N = mp(w_C)$, or N' is a nonterminal in G'_s with $w_N = mp(w_C)w_{N'}$.
7. $(A, A, N) \to Fail$ if $w_N \neq \varepsilon$.

– (T,C)-case. An illustration of the cases 4 to 6 is:



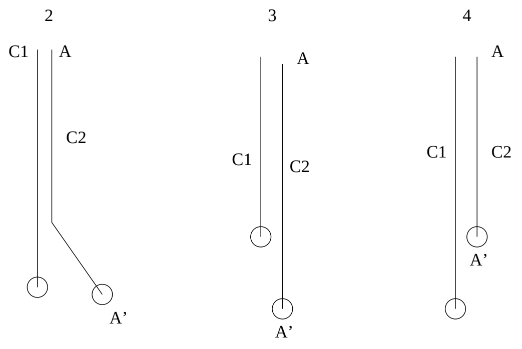
1. $(A, C, \varepsilon) \to \{(A_1, C, 1), \dots, (A_n, C, n),\}$ if $A ::= f(A_1, \dots, A_n)$.
2. $(A, C, \varepsilon) \to \{(C_2, C, \varepsilon), (A', C, N)\}$ if $A ::= C_2[A']$, where N is the nonterminal in G_s with $w_N = mp(w_{C_2})$.
3. $(A, C, N) \to (A, B_k, N')$ if $C ::= f(B_1, \dots, B_{i-1}, [\cdot], B_{i+1}, \dots B_n)$, and $w_N \perp mp(w_C)$, i.e. $k \neq i$, where N' is a nonterminal in G'_s with $w_N = kw_{N'}$.
4. $(A, C, N) \to (A, C_1, N)$ if $C ::= C_1 C_2$, $w_N \perp mp(w_{C_1})$.
5. $(A, C, N) \to (A, C_2, N')$ if $C ::= C_1 C_2$, and $mp(w_{C_1})$ is a prefix of w_N, where $N' = \varepsilon$ if $mp(w_{C_1}) = w_N$, or N' is a nonterminal in G'_s with $w_N = mp(w_{C_1})w_{N'}$.
6. $(A, C, N) \to \{(A, C_1, N), (C_2, A, N')\}$ if $C ::= C_1 C_2$, and w_N is a proper prefix of $mp(w_{C_1})$, where $w_N w_{N'} = mp(w_{C_1})$ and N' is a nonterminal in G'_s.

– (C,C)-case. An illustration of the cases 4 to 6:

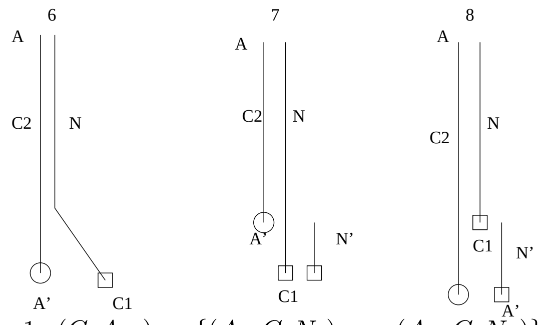


1. $(C_1, C_2, \varepsilon) \rightarrow \{(A_1, C_2, N_1), \ldots, (A_{i-1}, C_2, N_{i-1}), (A_{i+1}, C_2, N_{i+1}), \ldots, (A_n, C_2, N_n)\}$ if $C_1 ::= f(A_1, \ldots, A_{i-1}, [\cdot], A_{i+1}, \ldots, A_n)$, where G'_s contains the nonterminals $N_1, \ldots, N_n$ with rules $N_i ::= i$ for $i = 1, \ldots, n$.
2. $(C_1, C_2, \varepsilon) \rightarrow \{(C_3, C_2, \varepsilon), (C_4, C_2, N)\}$ if $C_1 ::= C_3C_4$, where N is the nonterminal in G_s with $w_N = mp(w_{C_3})$.
3. $(C_1, C_2, N) \rightarrow (C_1, A_k, N')$ if $C_2 ::= f(A_1, \ldots, A_{i-1}, [\cdot], A_{i+1}, \ldots A_n)$ and $w_N \perp mp(w_{C_2})$, i.e., $k \neq i$, where N' is a nonterminal in G'_s with $w_N = kw_{N'}$.
4. $(C_1, C_2, N) \rightarrow (C_1, C_3, N)$ if $C_2 ::= C_3C_4$, $w_N \perp mp(w_{C_3})$.
5. $(C_1, C_2, N) \rightarrow (C_1, C_4, N')$ if $C_2 ::= C_3C_4$, and $mp(w_{C_3})$ is a prefix of w_N, where $N' = \varepsilon$ if $mp(w_{C_3}) = w_N$, or N' is a nonterminal in G'_s with $w_N = mp(w_{C_3})w_{N'}$.
6. $(C_1, C_2, N) \rightarrow \{(C_1, C_3, N), (C_4, C_1, N')\}$ if $C_2 ::= C_3C_4$, and w_N is a proper prefix of $mp(w_{C_3})$, where $w_Nw_{N'} = mp(w_{C_3})$ and N' is a nonterminal in G'_s.

– (C,T)-case. An illustration of the transformations of ε-triples in cases 2 to 4 is:



An illustration of the transformation in cases 6 to 8 is:



1. $(C, A, \varepsilon) \to \{(A_1, C, N_1), \dots, (A_n, C, N_n)\}$ if $A ::= f(A_1, \dots, A_n)$, where G'_s contains the nonterminals $N_1, \dots, N_n$ with rules $N_i ::= i$ for $i = 1, \dots, n$.
2. $(C_1, A, \varepsilon) \to \{(C_1, C_2, \varepsilon), (A', C_1, N)\}$ if $A ::= C_2[A']$ and $mp(w_{C_1}) \perp mp(w_{C_2})$ where $w_N = mp(w_{C_2})$.
3. $(C_1, A, \varepsilon) \to (C_1, C_2, \varepsilon)$ if $A ::= C_2[A']$ and $mp(w_{C_1})$ is a prefix of $mp(w_{C_2})$.
4. $(C_1, A, \varepsilon) \to \{(C_1, C_2, \varepsilon), (A', C_1, N)\}$ if $A ::= C_2[A']$ and $mp(w_{C_2})$ is a proper prefix of $mp(w_{C_1})$, where N is the nonterminal in G_s with $w_N = mp(w_{C_2})$.
5. $(C, A, N) \to (C, A_i, N')$ if $A ::= f(A_1, \dots, A_n)$, where either $N' = \varepsilon$ if $w_N = i$, or N' is a nonterminal in G'_s with $w_N = iw_{N'}$.
6. $(C_1, A, N) \to (C_1, C_2, N)$ if $A ::= C_2[A']$ and $w_N \perp mp(w_{C_2})$.
7. $(C_1, A, N) \to \{(C_1, A', N')\}$ if $A ::= C_2[A']$ and $mp(w_{C_2})$ is a prefix of w_N, where either $N' = \varepsilon$ if $w_N = mp(w_{C_2})$, or N' is a nonterminal in G'_s with $w_N = mp(w_{C_2})w_{N'}$
8. $(C_1, A, N) \to \{(C_1, C_2, N), (A', C_1, N')\}$ if $A ::= C_2[A']$ and w_N is a proper prefix of $mp(w_{C_2})$, where N' is a nonterminal in G'_s with $w_N w_{N'} = mp(w_{C_2})$.

7.3 Failure Rules for Compaction

We will use periodicities of contexts for compaction of the set S_{ot}.
There are three failure rules, which we apply with high priority in the algorithm *WOPS*.

(CompFailAC) Given two different overlay triples $(A, C, N_1), (A, C, N_2)$ where A is a tree nonterminal and C is a context nonterminal. Let p_i be such that $mp(w_C) = w_{N_i} p_i$ for $i = 1, 2$. Fail, if $p_1 \perp p_2$.

(CompFailCCorth) Given two different orthogonal overlay triples $(D, C, N_1), (D, C, N_2)$, where D, C are context nonterminals, $|w_{N_1}| < |w_{N_2}|$, $mp(w_C) \perp w_{N_1} mp(w_D)$ and $mp(w_C) \perp w_{N_2} mp(w_D)$.
For $i = 1, 2$, let p_i be the maximal position, such that $w_{N_i} p_i$ is a prefix of $mp(w_C)$. For $i = 1, 2$, let q_i be the position, such that $w_{N_i} p_i q_i = mp(w_C)$.
Fail, if one of the following conditions hold:

– $p_1 \neq p_2$.
– $q_1 \perp q_2$

Correctness of the failure rule (CompFailCCorth) is proved in Lemma 9.7. Hence for orthogonal overlay triples (D, C, N) the part of the main path of D that is parallel to the main path of C is independent of N, and we can define the common path length (p_1 above) for orthogonal overlay triples $(D, C, \cdot)$ as $cp(D, C)$.

7.4 Compaction Rules

For the application of the compaction rules we assume that no failure rule applies. There are three different variants of the compaction rule for S_{ot}:

(CompactAC) Given three different tree overlay triples $(A, C, N_1), (A, C, N_2), (A, C, N_3)$ where A is a tree nonterminal and C is a context nonterminal, and $|w_{N_1}| < |w_{N_2}| < |w_{N_3}|$. If $(|w_{N_2}| - |w_{N_1}|) + (|w_{N_3}| - |w_{N_1}|) \leq |mp(w_C)| - |w_{N_1}|$, then replace the three triples by two triples $(A, C, N_1), (A, C, N_4)$, where N_4 is a nonterminal in G'_s and w_{N_4} is a prefix of $mp(w_C)$ such that $|w_{N_4}| = |w_{N_1}| + \gcd((|w_{N_2}| - |w_{N_1}|), (|w_{N_3}| - |w_{N_1}|))$.

(CompactCCpar) Given three different parallel overlay triples $(D, C, N_1), (D, C, N_2), (D, C, N_3)$ where D, C are context nonterminals, $mp(w_C) || w_{N_i} mp(w_D)$ for $i = 1, 2, 3$, and $|w_{N_1}| < |w_{N_2}| < |w_{N_3}|$. If $(|w_{N_2}| - |w_{N_1}|) + (|w_{N_3}| - |w_{N_1}|) \leq |mp(w_C)| - |w_{N_1}|$, then replace the three triples by two triples $(D, C, N_1), (D, C, N_4)$, where N_4 is a nonterminal in G'_s and w_{N_4} is a prefix of $mp(w_C)$ such that $|w_{N_4}| = |w_{N_1}| + \gcd((|w_{N_2}| - |w_{N_1}|), (|w_{N_3}| - |w_{N_1}|))$.

(CompactCCorth) Given four different orthogonal overlay triples $(D, C, N_0), (D, C, N_1), (D, C, N_2), (D, C, N_3)$ where D, C are context nonterminals, $mp(w_C) \perp w_{N_i} mp(w_D)$ for $i = 0, \ldots, 3$, and $|w_{N_0}| < |w_{N_1}| < |w_{N_2}| < |w_{N_3}|$. Let $p := |cp(D, C)|$ and let $d := |mp(w_C)| - p - 1$. If $(|w_{N_1}| - |w_{N_0}|) + (|w_{N_2}| - |w_{N_0}|) \leq (d - |w_{N_0}|)$, and $(|w_{N_2}| - |w_{N_1}|) + (|w_{N_3}| - |w_{N_1}|) \leq (d - |w_{N_1}|)$, then replace the two triples $(D, C, N_2), (D, C, N_3)$ by the triple (D, C, N_4), where N_4 is a nonterminal in G'_s and w_{N_4} is a prefix of $mp(w_C)$ such that $|w_{N_4}| = |w_{N_1}| + \gcd((|w_{N_2}| - |w_{N_1}|), (|w_{N_3}| - |w_{N_1}|))$.

Note that the difference between the three rules is in the kind of the nonterminal D, the relative position of the different D's in case D is a context nonterminal, and in the extra required triple (D, C, N_0) in rule (CompactCCorth).

Note also that it is possible for two context nonterminals C, D that there are orthogonal overlay triples as well as parallel overlay triples in S_{ot}.

8 Properties of the Transformation Rules

Lemma 8.1. *The simplification and transformation rules are sound and complete, i.e., if the state S is transformed into S', then*

1. *all the triples in S are well-formed iff all triples are well-formed in S'; and*
2. *all triples in S are valid iff they are valid in S'.*

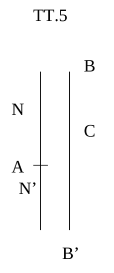
Proof. This follows by a straightforward, but tedious inspection of the rules. □

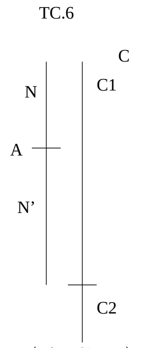
Now we count the the number of different triples during the transformation and the number of necessary transformations. The critical transformation rules are TT.5, TC.6, CC.6, and CT.8, since these rules increase the number of triples and can be applied more than once.

Lemma 8.2. *The transformation rules TT.5, TC.6, CC.6, and CT.8 have the following possibilities to generate new kinds of triples:*

1. *In case TT.5, with transformation $(A, B, N) \to \{(A, C, N), (B', A, N')\}$, the triple (A, C, N) is an overlay triple.*
2. *In case TC. 6, with transformation $(A, C, N) \to \{(A, C_1, N), (C_2, A, N')\}$, the triple (A, C_1, N) is an overlay triple.*
3. *In case CC.6, with transformation $(C_1, C_2, N) \to \{(C_1, C_3, N), (C_4, C_1, N')\}$, either (C_1, C_3, N) is an overlay triple, or (C_4, C_1, N') is immediately removed by simplification.*
4. *In case CT.8, with transformation $(C_1, A, N) \to \{(C_1, C_2, N), (A', C_1, N')\}$, either the triple (C_1, C_2, N) is an overlay triple, or the triple (A', C_1, N') is removed immediately by simplification.*

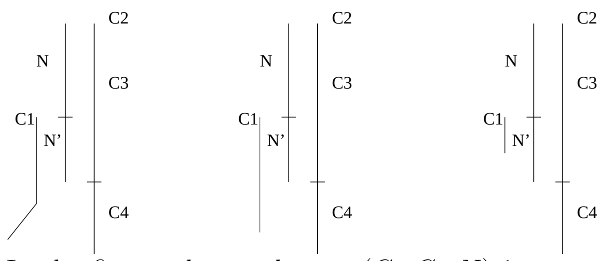
Proof. 1. The following picture illustrates the TT-case TT.5 and the TC-case TC.6:





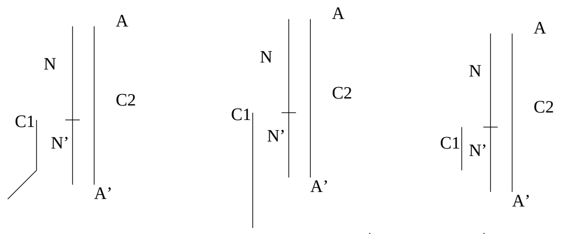
In the TT.5 it is obvious that (A, C, N) is an overlay triple, since A is a tree nonterminal and $w_N \parallel mp(w_C)$. In TC.6 case it is obvious that (A, C_1, N) is an overlay triple, since A is a tree nonterminal and $w_N \parallel mp(w_{C_1})$.

2. The following picture illustrates the three CC-cases corresponding to CC.6:



In the first and second case, (C_1, C_3, N) is an overlay triple, since $w_N \parallel mp(w_{C_3})$ and C_1 is a context nonterminal such that $w_N mp(w_{C_1})$ is not a prefix of $mp(w_{C_3})$. In the third case, (C_1, C_3, N) is an inclusion triple, and (C_4, C_1, N') will be removed by simplification , since $mp(w_{C_1})$ is a prefix of $w_{N'}$.

3. The following picture illustrates the three CT-cases corresponding to CT.8:



In the first and second case, (C_1, C_2, N) is an overlay triple, since $w_N \parallel mp(w_{C_2})$ and C_1 is a context nonterminal such that $w_N mp(w_{C_1})$ is not a prefix of $mp(w_{C_2})$. In the third case, (C_1, C_2, N) is an inclusion triple, and (A', C_1, N') will be removed by simplification , since $mp(w_{C_1})$ is a prefix of $w_{N'}$.

Lemma 8.3. *All strictly descending chains w.r.t. the strict ordering on triples have length at most* $2 * |G|$.

Proof. Obvious.

Lemma 8.4. *For every transformation of a triple* (D_1, D_2, N)*, the resulting triples are strictly smaller w.r.t. the ordering on triples defined in 6.4.*

Proof. Follows from an inspection of the rules.

Proposition 8.5. *For every transformation of a triple, the resulting triples are strictly smaller.*

For every transformation of a non-ε-triple, after applying simplification to the resulting triples, at most one triple of the result is not an overlay triple.

Lemma 8.6. *Let G be the initial STG.*
The number of ε-triples is at most $|G|^2$.
The number of non-ε-triples coming from a single transformation from ε-triples is $\leq (|G|^2) * maxarity$.

Proof. The upper bound is obvious. The treatment of ε-triples in the transformations shows that at most *maxarity* triples may arise from a single ε-triple.

Lemma 8.7. *Let G be the initial STG. During every run of the algorithm WOPS, the number of triples in S_{not} is bounded by maxarity.*

Proof. Only ε-triples may add *maxarity* triples to S_{not}. The priority of choosing triples for further transformations and Proposition 8.5 show that the focussed triple may result in at most one further triple from S_{not}, and perhaps triples in the other sets.

Since we already have bounded the sets of ε-triples and the set S_{not}, it remains to treat the set S_{ot}.

9 Correctness of the Compaction and Failure Rules

For the proofs of correctness we assume that the simplifications are exhaustively applied.

9.1 The tree-context-case

The following lemma is also proved in [SSS98].

Lemma 9.1. *Let W be a nontrivial context, and let p be a position, such that W overlaps itself at position p, i.e., for every position q, such that $pq \in Pos(W)$, we have $head(W|_{pq}) = head(W|_q)$. Then the following holds:*

1. *$p \parallel mp(W)$, i.e. the overlapping start on the main path.*
2. *$mp(W) \parallel p \cdot mp(W)$, i.e., the paths of both occurrences of W are on the same path.*

Proof. The first claim is clear, since $p \perp mp(W)$ would imply that W contains W properly.

In the second case, assume that $mp(W) \perp p \cdot mp(W)$. Then $p \neq \varepsilon$. Let p' be the maximal prefix of $mp(W)$, such that $pp' \parallel mp(W)$. Then $W|_{pp'} = f(t_1, \ldots, t_{i-1}, \cdot, t_{i+1}, \ldots, t_n)$, and there is $k \neq i$, such that $p'k$ is a prefix of $mp(W)$. But then t_k contains itself properly, a contradiction.

Lemma 9.2. *The failure rule (CompFailAC) is correct.*

Proof. Assume we have two valid overlay triples $(A, C, N_1), (A, C, N_2)$, where A is a tree nonterminal and C is a context nonterminal. Since $N_1 || N_2$, we can assume w.l.o.g. that $|w_{N_1}| < |w_{N_2}|$.
Let p_i be such that $mp(w_C) = w_{N_i} p_i$ for $i = 1, 2$. Let W be the context derived from w_A by plugging in the hole at position p_1. Since (A, C, N_1) is a overlay triple, we also have $W = w_C|_{w_{N_1}}$. The valid overlay triple (A, C, N_2) with $|w_{N_1}| < |w_{N_2}|$ implies that there is another overlay of W in w_C starting at position w_{N_2}. Now Lemma 9.1 shows that $p_1 || p_2$.

Lemma 9.3. *The compaction rule (CompactAC) is sound and complete.*

Proof. Assume we have three valid overlay triples $(A, C, N_1), (A, C, N_2), (A, C, N_3)$, where A is a tree nonterminal and C is a context nonterminal, the failure rules do not apply, and $|w_{N_1}| < |w_{N_2}| < |w_{N_3}|$. Let p_i be such that $mp(w_C) = w_{N_i}p_i$ for $i = 1, 2, 3$. Since the failure rules do not apply, p_3 is a prefix of p_2 and p_2 is a prefix of p_1. Thus we have an overlapping situation for contexts (similar to words). Let $W := w_C|_{w_{N_1}}$. The two triples $(A, C, N_1), (A, C, N_2)$ show that W has a period $|w_{N_2}| - |w_{N_1}|$, the two triples $(A, C, N_1), (A, C, N_3)$ show that W has a period $|w_{N_3}| - |w_{N_1}|$, and the context W seen as word, has length $|mp(w_C)| - |w_{N_1}|$. Hence we can use Lemma 2.1, and obtain that if $(|w_{N_2}| - |w_{N_1}|) + (|w_{N_3}| - |w_{N_1}|) \leq |mp(w_C)| - |w_{N_1}|$, the context W also has period $\gcd((|w_{N_2}| - |w_{N_1}|), (|w_{N_3}| - |w_{N_1}|))$. Thus the generation of N_4 and the replacement of $(A, C, N_2), (A, C, N_3)$ with (A, C, N_4) keeps exactly the information.

9.2 The parallel context-context-case

Similar to the tree-context-case.

Lemma 9.4. *The compaction rule (CompactCCpar) is sound and complete.*

Proof. The proof is the same as the proof of Lemma 9.3 by replacing W with W', where $W' = w_D|_p$, and $w_{N_1}p = mp(w_C)$.

9.3 The orthogonal context-context-case

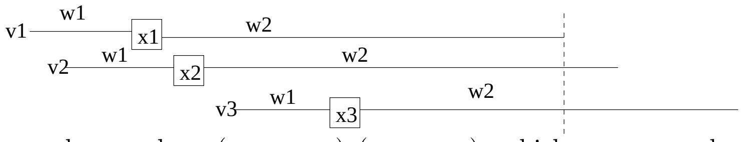
We need a helpful lemma on overlaps of a word with itself, where one position is undefined, before we can treat the orthogonal CC-case. It can be seen as a generalization of a periodicity lemma for a word overlapping with itself to a periodicity lemma of a word w overlapping with its single-point mutants where the mutations are at a fixed position in w.

We denote the letter at position i in a word with $w[i]$, where we assume that the first letter is $w[1]$. With (v, w, k) we denote an overlap of the words v, w where w starts with the $(k + 1)^{\text{st}}$ letter of v. The notation (v, w, k) means that for all $i = k+1, \dots, min(|v|, |w|+k)$, we have $v[i] = w[i-k]$. The triple (v, v, k) implies that the word v has period k.

Lemma 9.5. *Let w_1, w_2 be words, x_1, x_2, x_3 be letters, and $v_i = w_1x_iw_2$ for $i = 1, 2, 3$. Let there be two overlaps $(v_1, v_2, m), (v_1, v_3, n)$ for $1 \leq m < n$. Then the following holds:*

If $m + n \leq |w_2|$, then v_2 has period $\gcd(m, n)$, $x_2 = x_3 = w_2[\gcd(m, n)]$, and $v_2 = v_3$.

Proof.



There are also overlaps $(w_2, w_2, m), (w_2, w_2, n)$, which means w_2 has periods m, n. Hence by the condition $m + n \leq |w_2|$ and Lemma 2.1, $r := \gcd(m, n) \leq min(m, n)$ is also a period of w_2. The overlap shows (see e.g. the illustration), that $x_2 = w_2[m]$. Comparing v_1, v_2 and from the period r, we obtain $x_2 = w_2[m] = w_2[m + h * r]$ for integers h, as long as $1 \leq m + h * r \leq |w_2|$, hence $x_2 = w_2[r]$. The same holds for the comparison of v_1, v_3, which implies $x_2 = x_3 = w_2[r]$, and also $v_2 = v_3$. There is an overlap (v_3, w_2, n), and the overlap is synchronized, since $m + n \leq |w_2|$ and the w_2-parts of v_1, v_3 have an overlap larger than r. Let w_3 be the maximal suffix of length $\leq n$ of w_1. Then the overlap of v_1 with v_3 shows that $w_3x_3w_2$ is a suffix of v_3 and has period r. If $|w_1| < m$, then $w_3x_3w_2 = v_2 = v_3$, and v_2, v_3 have period r. If $|w_1| \geq m$, then v_2 and x_1 in v_1 have a common position in the overlap of v_1, v_2. Due to the period r, we obtain that $x_1 = x_2$.

Example 9.6. There are examples with $x_1 \neq x_2$ in the first case of Lemma 9.5; e.g.: $w_1 = \mathtt{a}, w_2 = \mathtt{ababab}, x_1 = \mathtt{c}, x_2 = x_3 = \mathtt{b}$, $m = 2, n = 4$. We use $x = x,y = x_2, z = x_3$. The values of y, z are fixed to be b, but the value of x is not restricted.

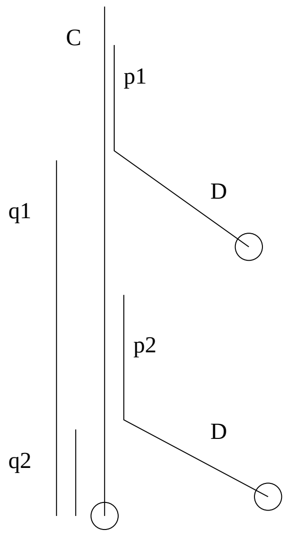
v_3	`. . . . a z a b a b a b a b a b`
v_1	`a x a b a b a b a b a b`
v_2	`. . a y a b a b a b a b a b`

Lemma 9.7. *Let there be two valid orthogonal overlay triples $(D, C, N_1), (D, C, N_2)$, where C, D are context nonterminals, such that $w_{N_1} mp(w_D) \perp mp(w_C)$ and $w_{N_2} mp(w_D) \perp mp(w_C)$. For $i = 1, 2$ let p_i be the maximal paths such that p_i is a prefix of w_D and $w_{N_i} p_i \parallel mp(w_C)$, and let q_i be such that $w_{N_1} p_i q_i = mp(w_C)$. Then the following holds:*

– $p_1 = p_2$.

– $q_1 \parallel q_2$.

A picture of a possible situation is:



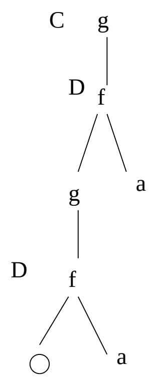
Proof. W.l.o.g. we assume that $|w_{N_1}| < |w_{N_2}|$.
First suppose that $p_1 \neq p_2$. From the assumptions it is clear that $p_1 \parallel p_2$. Let q be such that $w_{N_1}p_1q = w_{N_2}p_1$. Looking at the picture, we have to distinguish the cases that p_1 is a prefix of p_2, and the case that p_2 is a prefix of p_1. It is easy to see that in each of the cases we can construct an arbitrarily long path in D of the form p_1q^+, which contradicts the finiteness of contexts. This proves the first part.

Now assume that $q_1 \perp q_2$. Let $w_D|_{p_1} = f(t_1, \ldots, t_{i-1}, W', t_{i+1}, \ldots, t_n)$ and let k be the first letter of w_{q_1}. It is clear that $k \neq i$. Let D' be generated from $D|_{p_1k}$ by plugging in a hole at position q_1. Now we can apply Lemma 9.1 to the overlap of D' with itself and obtain that $q_1 \parallel q_2$.

Example 9.8. We give an example that the orthogonal case treated in Lemma 9.7 may indeed be possible.

$$\begin{array}{llll}
C & := g(f(g(f([\cdot], a)), a)) & p_1 & := \varepsilon \\
D & := f(g(f(a, a)), [\cdot]) & p_2 & := \varepsilon \\
w_{N_1} & := 1 & q_1 & := 1.1.1 \\
w_{N_2} & := 1.1.1 & q_2 & := 1
\end{array}$$

A picture of the overlapping situation is:



Lemma 9.9. *The failure rule (CompFailCCorth) is correct.*

Proof. Follows from Lemma 9.7.

Lemma 9.10. *The compaction rule (CompactCCorth) is sound and complete.*

Proof. Recall the notation of the rule (CompactCCorth). For $i = 0, \ldots, 3$, let k_i be a letter and r_i be a postion string, such that $\mp(C) = w_{N_i} \cdot cp(D, C) \cdot k_i \cdot r_i$, where k_i is a letter. For $i = 0, 1, 2, 3$, let $u_{1,i}, x_i, u_{2,i}$ be contexts, such that $u_{1,i}x_iu_{2,i} = (w_C)_{|N_i}$, and $mp(u_{1,i}) = cp(D, C)$. Lemma 9.7 can be applied to show that $u_{1,0} = u_{1,1} = u_{1,2} = u_{1,3}$ and that $u_{2,3}$ is a prefix of $u_{2,2}$, $u_{2,2}$ is a prefix of $u_{2,1}$, and $u_{2,1}$ is a prefix of $u_{2,0}$. Now we can apply Lemma 9.5 to the words $v_0 := u_{1,0}x_0u_{2,0}$ and $v_1 := u_{1,0}x_1u_{2,0}$ and $v_2 := u_{1,0}x_2u_{2,0}$, which due to the first length condition in the rule implies that $x_1 = x_2$ and $v_1 = v_2$. A further application of Lemma 9.5 to the words $v_1' := u_{1,0}x_1u_{2,1}$ and $v_2' := u_{1,0}x_2u_{2,1}$ and $v_2 := u_{1,0}x_3u_{2,1}$ shows that $x_2 = x_3$ and $v_2' = v_3'$ due to the second length condition in the rule. This implies that $v_1' = v_2' = v_3'$, and we can use the periodicity Lemma 9.5 and Lemma 9.7 can be applied. The length condition on N_0, N_1, N_2 in rule (CompactCCorth) implies by Lemma 9.5 (by appropriately prolonging that v_i' is a suffix of $w_{2,1}$ and the length condition on N_1, N_2, N_3 in rule (CompactCCorth) implies that $w_{2,3}$ is a suffix of $w_{2,2}$. Thus we can use periods of words as in the parallel case (see the proof of Lemma 9.3), and no information is lost by replacing the two triples by the N_4-triple.

9.4 Restricting the Number of Overlay Triples

We generalize Plandowski's argument to show that there will be at most $O(n^3)$ overlay triples for (A, C, N).

Lemma 9.11. *If failure and compaction rules have been exhaustively applied, then the cardinality of S_{ot} is bounded by $(5n + 4) * n^2$, where $n = |G|$, i.e. by $O(n^3)$.*

Proof. The proof follows Plandowski's reasoning [Pla94]:
We fix D, C and assume that there are overlay triples in S_{ot}. There may be parallel overlay triples and orthogonal overlay triples in S_{ot}. First we consider sequences of triples that consist only of parallel overlay triples. Let $(D, C, N_1), (D, C, N_2), (D, C, N_3), \ldots, (D, C, N_k)$ be triples in S_{ot}, where we assume that $|w_{N_1}| < |w_{N_2}| < \ldots < |w_{N_k}|$. Let j be an index in the sequence. To simplify the reasoning, let $d_i := |w_{N_i}|$ for all appropriate i. Since compaction is not applicable, the inequation $(d_{j+2} - d_j) + (d_{j+1} - d_j) > |mp(w_C)| - d_j$ holds. Since $d_{j+2} > d_{j+1}$, this implies $2 * d_{j+2} - d_j > |mp(w_C)|$. This gives $|mp(w_C)| + 2 * d_{j+2} - d_j > 2 * |mp(w_C)|$ which implies $|mp(w_C)| - d_j > 2(|mp(w_C)| - d_{j+2})$ and $0.5(|mp(w_C)| - d_j) > |mp(w_C)| - d_{j+2}$. Since $|mp(w_C)| \leq 2^{|G|}$, and the numbers on the left and right of the inequation are positive, the sequence of numbers $|mp(w_C)| - d_1, |mp(w_C)| - d_2, \ldots, |mp(w_C)| - d_k$ has at most $2|G| + 1$ elements.

Now we can also treat sequences of purely orthogonal overlay triples. The difference is that given a sequence $(D, C, N_1), (D, C, N_2), (D, C, N_3), \ldots$, the same estimation as above shows that for every$j$, provided the indices $j+1, j+2, j+3$ are valid, the inequation $0.5(d - d_j) > d - d_{j+2}$ or $0.5(d - d_{j+1}) > d - d_{j+3}$ holds, where $d := |mp(w_C)| - |cp(D, C)| - 1$. It is clear that $d - d_k \geq 0$ for all k must hold. The estimation shows that the possible interval length $(d - d_k)$ is divided at least by two going from j to $j + 3$, hence there are at most $3|G| + 3$ elements in a sequence d_i. Since per pair D, C there may be 2 kinds of sequences, the overall maximal number is $(5n + 4) * n^2$, where $n = |G|$.

9.5 Complexity of *WOPS*

Lemma 9.12. *The size of the position grammar in WOPS is bounded by $|G| + P * depth(G)$, where P is the number of transformation and compaction steps of WOPS.*

Proof. The size-increase of G_s in every step is always the addition of a prefix or suffix of a position of a certain length. This may happen in the transformation rules or in the compaction rules. By Lemma 3.6 the size increase in every step is at most by adding $depth(G)$.

Now we can prove the main theorem:

Theorem 9.13. *The algorithm WOPS requires at most polynomially many steps in the size $|G|$ of the input STG G.*

Proof. For the complexity of the algorithm we have to take into account the priority of the rule applications. The sequence of states can be divided into intermediate states (we call them *pot-states*) where only $S_{\varepsilon,closed}$ and S_{ot} are nonempty and failure and compaction rules are applied exhaustively. The transformation sequence between two such pot-states is called pot-meta-transformation. We have to argue that only polynomially many pot-states are possible.

The first argument is that S_{ot} itself is polynomially bounded by $O(|G|^3)$, which is proved in Lemma 9.11. Let (D, C, N) be a maximal triple in S_{ot}. Then there are at most $5|G|+4$ triples of the form (D, C, N') in S_{ot}. Thus after at most $5|G|+4$ pot-meta-transformations of S_{ot}, a pot-state is reached, such that it is no longer possible to have a triple (D, C, N') in any active set in any successor state. This can happen at most $|G|^2$ times. Thus the number of pot-meta-transformations is $O(|G|^3)$, i.e., polynomial.

The number of transformations and compactions in a single pot-meta-transformation can be bounded above as follows. There may be at most $O(|G|^2)$ triples in $S_{\varepsilon,open}$, and the immediate successor triples are at most $O(|G|^2)$ (see Lemma 8.6). Every triple may give rise to a transformation of length at most $O(|G|^2)$. Every step may be accompanied by polynomially many compaction rule applications. Hence the number of transformations and compaction rule applications in a pot-meta-transformation is polynomially bounded. Lemma 9.12 shows that the size of the position grammar is polynomially bounded, since we have at most polynomially many transformations and compactions. It is now easy to see that a single rule application: transformation, simplification, compaction, and compaction failure rules, can be done in polynomial time, hence a pot-meta-transformation can be performed in polynomial time.

We conclude that the algorithm *WOPS* requires polynomial time depending on $|G|$.

The maximal exponent in the polynomial is at least 6, however, a thorough estimation of the exponent is not possible, since I did not find any estimation in the literature of the degree of the polynomial of the worst case running timer of Plandowski's algorithm for context free grammars.

Corollary 9.14. *The word problem for tree and context nonterminals in STGs is in P.*

Proof. For tree nonterminals, this follows from 9.13. For context nonterminals C_1, C_2 we extend the grammar by new tree nonterminals and the rules $A_1 := C_1[a], A_2 := C_2[a]$, where a is a fresh constant. Now Theorem 9.13 shows that the complexity is polynomial.

Potential Optimizations The algorithm *WOPS* has a potential for optimizations, which we will not include, e.g. the following rules would be advantageous.

- Do not expand (D, D, ε), where D is a tree or context nonterminal. I.e., move this triple immediately to $S_{\varepsilon,closed}$.
- Fail if there is (A, A, N), where A is a tree nonterminal, $w_N \neq \varepsilon$.
- Fail if there is (C, C, N), where C is a tree nonterminal, and $mp(w_C) \perp w_N$.

10 Conclusion and Future Work

It is shown that in a sharing representation of terms, sharing subterms and unary contexts where also composition of contexts is permitted, it is possible to compare the represented terms for equality in polynomial time. The base are singleton tree grammars with (unary) contexts as a generalization of Plandowski's singleton grammars.

Applications of this work may be in determining an improved upper bound of the complexity of stratified context unification and bounded second order unification, which are conjectured to be NP-complete. More practical applications may be in implementations of Automated Deduction Systems using a term representations with exponents, or using a representation with sharing of terms and contexts and exploiting the polynomial equality test to ensure a unique representation.

Future work is to exhibit the smallest degree of the necessary polynomial in the complexity and to optimize the algorithm; this, however, requires to first exhibit the degree of the worst case running time of Plandowski's algorithm. Another direction of future work is to investigate whether the polynomial complexity also holds if contexts with multiple holes or if nonlinear term functions instead of contexts are used in the representation.

Another line of research is to investigate the word problem for unordered, labelled trees.

11 Acknowledgements

I thank David Sabel for reading a draft of this paper, and Klaus Schulz for his comments on a draft of this paper.

References

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 October, 1rst 2002.

CH95. Hong Chen and Jieh Hsiang. Recurrence domains: Their unification and application to logic programming. *Information and Computation*, 122(1):45–69, 1995.

Com95. Hubert Comon. On unification of terms with integer exponents. *Mathematical Systems Theory*, 28(1):67–883, 1995.

GNN04. Harald Ganzinger, Robert Nieuwenhuis, and Pilar Nivela. Fast term indexing with coded context trees. *J. of Automated Reasoning*, 32:103–120, 2004.

HG97. Miki Hermann and Roman Galbavý. Unification of infinite sets of terms schematized by primal grammars. *Theor. Comput. Sci.*, 176(1–2):111–158, 1997.

HS98. Miki Hermann and Gernot Salzer. On the word, subsumption, and complement problem for recurrent term schematizations. In *MFCS 19989*, volume 1450 of *Lecture Notes in Computer Science*, pages 257–266, 1998.

Lot83. M. Lothaire, editor. *Combinatorics on words.* Cambridge University Press, 1983.

LSSV04. Jordi Levy, Manfred Schmidt-Schauß, and Matteu Villaret. Monadic second-order unification is NP-complete. In *Rewriting Techniques and Applications (RTA-15)*, LNCS 3091, pages 55–69. Springer, 2004.

MM82. Alberto Martelli and Ugo Montanari. An efficient unification algorithm. *ACM Transactions on Programming Languages and Systems*, 4(2):258–282, 1982.

Pla94. Wojciech Plandowski. Testing equivalence of morphisms in context-free languages. In *ESA 94*, volume 855 of *Lecture Notes in Computer Science*, pages 460–470, 1994.

Rob65. J.Alan Robinson. A machine oriented logic based on the resolution principle. *J. of the ACM*, 12(1):23–41, 1965.

Ryt04. Wojciech Rytter. Grammar Compression, LZ-encodings, and string algorithms with implicit input. In J. Diaz et. al., editor, *ICALP 2004*, volume 3142 of *LNCS*, pages 15–27. Springer-Verlag, 2004.

Sal92. Gernot Salzer. The unification of infinite sets of terms and its applications. In *LPAR 1992*, volume 624 of *Lecture Notes in Computer Science*, pages 409–420, 1992.

SS01. Manfred Schmidt-Schauß. Stratified context unification is in PSPACE. In *Proceedings of CSL'01*, LNCS 2142, pages 498–512, 2001.

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

SS04. Manfred Schmidt-Schauß. Decidability of bounded second order unification. *Information and Computation*, 188(2):143–178, 2004.

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