Fundamenta Informaticae 29 (1997) 1–26
IOS Press

ON USING SEMI-DYCK SETS TO ANALYSE COUPLED-CONTEXT-FREE LANGUAGES

Günter HOTZ and Gisela PITSCH*
FB 14 - Informatik, Universität des Saarlandes, D-66123 Saarbrücken, Germany,
e-mail: hotz/pitsch@cs.uni-sb.de

Abstract. Coupled-Context-Free Grammars are a natural generalization of context-free grammars obtained by combining nonterminals to corresponding parentheses which can only be substituted simultaneously. Refering to their generative capacity we obtain an infinite hierarchy of languages that comprises the context-free ones as the first and all those generated by Tree Adjoining Grammars (TAGs) as the second element. The latter is important because today, TAGs are commonly used to model the syntax of natural languages. Here, we present a completely new approach to analyse this language hierarchy. It solves the word problem for the class of languages generated by TAGs in time $O(n^6)$, n length of the input, by reducing it to the analysis of sequences of parentheses.

1. Introduction

In order to analyse natural languages, the first problem to be solved is to model the syntax formally. Many investigations, as for example [20], show that this cannot be done by context-free grammars. For context-sensitive grammars which are powerful enough it is known that they are PSPACE-complete. Thus, there is a trade-off between the power of the formalism and its analysis complexity. To solve this dilemma, people are looking for language classes completely in between context-free and context-sensitive languages that are powerful enough to model the syntax of natural languages but showing a polynomial time complexity as to the analysis. Coupled-Context-Free Grammars represent such a formalism. Their suitability to model syntactical phenomena follows from the fact that they include the languages generated by the Tree Adjoining Grammars (TAGs) of [12] as one subclass. The linguistic significance of TAGs is well explored (cf. [1], [13], [14], e.g.). In particular and among other properties, both formalisms are able to model the linguistic phenomenon of cross-serial dependencies, which is not context-free but frequently appears in natural languages (cf. [2], [20]).

*This research has been supported by a Graduiertenkolleg-fellowship of the Deutsche Forschungsgemeinschaft and by the ESPRIT Basic Research Action No. 6317 (ASMICS2).

The formalism of Coupled-Context-Free Grammars has been introduced in [6], [7], and further studied in [9], [10], [16], [17]. It belongs to the family of regulated string rewriting systems investigated in [4]. Many regulated string rewriting systems as, e.g., the related Scattered Context Grammars of [5] obtain a generalization of context-free grammars by allowing simultaneous rewriting of arbitrary finite combinations of elements. In [4], it is shown that this results in languages which are not semilinear. But semilinearity is important since it formalizes the "constant-growth property" of natural languages. In contrast to these formalisms, all languages defined by Coupled-Context-Free Grammars are semilinear because of two restrictions on the simultaneous rewriting steps. First, only those elements can be rewritten simultaneously which were produced by the same rewriting step. Second, the Coupled-Context-Free Grammars consider elements rewritten simultaneously as components of a parenthesis. Those can only be substituted if they form a parenthesis and only by sequences of parentheses which are correctly nested.

When characterizing the Coupled-Context-Free Grammars by the maximal number of elements rewritten simultaneously – the *rank* of the grammar – we get an infinite hierarchy, whose smallest element – the one of rank 1 – is represented by the context-free grammars. The next element, namely Coupled-Context-Free Grammars of rank 2, generates the same class of languages as the Tree Adjoining Grammars of [11] and [12]. This growth of the generative capacity permits to extend the context-free languages by continuously growing semilinear language classes. Thus, Coupled-Context-Free Grammars represent one possibility to characterize more precisely the gap between context-free and context-sensitive languages.

Since Coupled-Context-Free Grammars are a natural generalization of context-free ones, all efficient context-free parsing algorithms represent possible starting points to develop such an algorithm for them. Thus, the well-known algorithms of Earley resp. Younger were used to develop algorithms to analyse TAGs (cf. [21] resp. [22]) and Coupled-Context-Free Grammars of rank 2 (cf. [7] resp. [9]). But all of them are essentially based on the context-free counterpart, not on the special features of the formalisms. Therefore, to get a lower time complexity of the analysis, we should try to exploit these special features.

Coupled-Context-Free Grammars are defined via expressions over sets of parentheses. Therefore, a natural way to solve their word problem seems to be the reduction of the problem to the investigation of sequences of parentheses. This is possible, if we are able to translate any combination of input word/Coupled-Context-Free Grammar into a set of parentheses expressions which uniquely represents this instance of the general word problem. Here, we present the first realisation for his approach.

In the context-free case, the representation theorem of Shamir (cf. [19]) can be used to construct a directed graph for any combination of input word/grammar in Greibach normalform. Its edges are marked such that the input word is an element of the language generated by this grammar if and only if there exists a path from the distinguished begin node to the distinguished end node of the graph which is marked by a sequence of parentheses correctly nested.

Based on the context-free case, we construct such a graph for Coupled-Context-Free Grammars of rank 2. But here, its edge marking consists of two components. The first one stores the information about context-free derivability as before, the second one stores the information about the coupling between two components of a parenthesis. Now, we aim to find a path through the graph whose marking represents a correct sequence of parentheses in the first as well as in the second component. The direct translation of the context-free algorithm onto our situation leads to the solution of an NP-complete problem as shown in the Appendix. Therefore, we have to develop an alternative strategy. It looks at first on the correctness of the coupling, i.e. on the

second component of the marking. Only then, the context-free derivability is tested relative to this coupling, i.e. the first component is investigated.

The complexity of the algorithm resulting for Coupled-Context-Free Grammars of rank 2 amounts to $O(n^6)$, where n is the length of the input. This equals the time bound as it is achieved by [21] or [9]. But since it mainly relies on the special features of the formalism, it represents the best starting point for future research. Besides, the algorithm provides new insights on the general structure of these grammars which already led to a representation theorem for them in [10].

The paper starts by defining Coupled-Context-Free Grammars. Then, the context-free case is recalled. This includes the representation theorem of Shamir, the construction of the graph, and the analysis. Subsequently, we develop the main ideas our algorithm is based on before presenting the algorithm itself. Finally, we discuss its correctness and complexity. The paper ends by the proof of the NP-completeness of a related, slightly more general analysis problem on directed, marked acyclic graphs.

2. Coupled-Context-Free Grammars

In this section, we present in detail the formalism of Coupled-Context-Free Grammars. An extensive characterization can be found in [7] and [6].

Coupled-Context-Free Grammars are defined over extended semi-Dyck sets, a generalization of semi-Dyck sets. Elements of these can be regarded as sequences of parentheses that are correctly nested. Semi-Dyck sets play an important role in the theory of formal languages. [3] show that they are a generator of context-free languages. To extend the family of context-free languages using them we consider parentheses of arbitrary finite order and Extended Semi-Dyck sets over them defined as:

Definition 2.1. (Parentheses Set)
A finite set $\mathcal{K} := \{(k_{i,1}, \ldots, k_{i,m_i}) \mid i, m_i \in \mathbf{N}\}$ *is a* Parentheses Set *if and only if it satisfies*

$$k_{i,j} \neq k_{l,m} \quad \text{for } i \neq l \text{ or } j \neq m.$$

The elements of $\mathcal{K}$ *are called* Parentheses. *All parentheses of a fixed length* $r \geq 1$ *are summarized as*

$$\mathcal{K}[r] := \{(k_{i,1}, \ldots, k_{i,m_i}) \in \mathcal{K} \mid m_i = r\}$$

where $\mathcal{K}[0] := \{\varepsilon\}$. ($\varepsilon$ *is the empty word.*) *The set of all components of parentheses in* $\mathcal{K}$ *is denoted by*

$$comp(\mathcal{K}) := \{k_i \mid (k_1, \ldots, k_i, \ldots, k_r) \in \mathcal{K}\}.$$

Notation 1 *When dealing only with parentheses of length at most 2 (the situation we are used to), we denote pairs of parentheses by* $(X, \overline{X})$, $(Y, \overline{Y})$, ... *instead of* (X_1, X_2), (Y_1, Y_2), ... *This simplifies the formulas since we can use* X_1, X_2, ... *for other purposes.*

Definition 2.2. (Extended Semi-Dyck Set)
Let $\mathcal{K}$ *be a parentheses set,* T *an arbitrary finite set where* $T \cap \mathcal{K} = T \cap comp(\mathcal{K}) = \emptyset$. $ED(\mathcal{K}, T)$, *the* Extended Semi-Dyck Set over $\mathcal{K}$ and T, *is inductively defined by*

($E1$) $T^* \subseteq ED(\mathcal{K}, T)$.
($E2$) $\mathcal{K}[1] \subseteq ED(\mathcal{K}, T)$.
($E3$) $u_1, \ldots, u_r \in ED(\mathcal{K}, T), (k_1, \ldots, k_{r+1}) \in \mathcal{K}[r+1] \Longrightarrow k_1 u_1 \cdots u_r k_{r+1} \in ED(\mathcal{K}, T)$.
($E4$) $u, v \in ED(\mathcal{K}, T) \Longrightarrow u \cdot v \in ED(\mathcal{K}, T)$.
($E5$) $ED(\mathcal{K}, T)$ *is the smallest set fulfilling conditions* $(E1) - (E4)$.

Now, we are able to define our grammars.

Definition 2.3. (Coupled-Context-Free Grammar)
A Coupled-Context-Free Grammar over $ED(\mathcal{K},T)$ *is an ordered 4-tuple* $G=(\mathcal{K},T,P,S)$ *where* $S\in\mathcal{K}[1]$ *holds and where* P *is a finite, nonempty set of productions of the form*

$$\{(k_1,\ldots,k_r)\to(\alpha_1,\ldots,\alpha_r)\mid(k_1,\ldots,k_r)\in\mathcal{K},\alpha_1\cdot\ldots\cdot\alpha_r\in ED(\mathcal{K},T)\}.$$

The set of all these grammars is denoted by $CCFG$.

The term "coupled" expresses that here, a certain number of context-free rewritings is executed in parallel and controlled by $\mathcal{K}$. Therefore, $\mathcal{K}$ can be regarded as a set of coupled nonterminals.

To simplify the notation, we denote the left resp. right side of $p:=(k_1,\ldots,k_r)\to(\alpha_1,\ldots,\alpha_r)\in P$ by

- $\mathcal{S}(p):=(k_1,\ldots,k_r)$, the *source* of p, resp.
- $\mathcal{D}(p):=(\alpha_1,\ldots,\alpha_r)$, the *drain* of p.

The same abbreviation is used for any edge $e=(v_1,v_2)\in E$ in a directed graph $G=(V,E)$ to denote the source ($\mathcal{S}(e)=v_1$) resp. the drain ($\mathcal{D}(e)=v_2$) of e.

At last, we give the definition of derivation for $CCFG$. Let $G=(\mathcal{K},T,P,S)\in CCFG$ and $V:=comp(\mathcal{K})\cup T$. We define the relation $\Rightarrow_G$ as a subset of $V^*\times V^*$ consisting of all *derivation steps of rank* r for G with $r\geq 1$. $\phi\Rightarrow_G\psi$ holds for $\phi,\psi\in V^*$ if and only if there exist $u_1,u_{r+1}\in V^*$, $u_2,\ldots,u_r\in ED(\mathcal{K},T)$ and $(k_1,\ldots,k_r)\to(\alpha_1,\ldots,\alpha_r)\in P$ such that

$$\phi=u_1k_1u_2k_2\cdots u_rk_ru_{r+1}\quad\text{and}\quad\psi=u_1\alpha_1u_2\alpha_2\cdots u_r\alpha_ru_{r+1}.$$

$\overset{*}{\Rightarrow}_G$ denotes the reflexive, transitive closure of $\Rightarrow_G$. Obviously, for the above ϕ and ψ, $u_1\cdot u_{r+1}\in ED(\mathcal{K},T)$ follows from $S\overset{*}{\Rightarrow}_G\phi$ since the result of the substitution is a sequence of parentheses correctly nested if and only if the original word was. A sequence $\phi_1,\ldots,\phi_n$ with $\phi_i\Rightarrow_G\phi_{i+1}$ for all $1\leq i<n$ and $\phi_1=\phi$, $\phi_n=\psi$ is called a *derivation of* ψ *from* ϕ *in* G. The language generated by G is defined as

$$L(G):=\{w\in T^*\mid S\overset{*}{\Rightarrow}_G w\}.$$

In order to be able to describe the generative capacity of Coupled-Context-Free Grammars of different ranks exactly, we need the following notions:

Definition 2.4. (Rank, $CCFG(l)$)
For any $G=(\mathcal{K},T,P,S)\in CCFG$, *let the* rank *of* G *be defined as* $rank(G):=\max\{r\mid(k_1,\ldots,k_r)\in\mathcal{K}\}$. *Then, we define* $CCFG(l):=\{G\in CCFG\mid rank(G)\leq l\}$ *for all* $l\geq 1$.

The following theorem proven in [7] shows that $CCFG$ builds up an infinite hierarchy of languages and, at the same time, represents a proper extension of context-free grammars not exceeding the power of context-sensitive grammars:

Theorem 2.1. [Hierarchy]
Let CFL be the family of all context-free, CSL the family of all context-sensitive languages, TAL the family of all languages generated by TAGs, and $CCFL(l)$ the one generated by $CCFG(l)$. It holds:

(1) $CFL=CCFL(1)$, $TAL=CCFL(2)$.
(2) $CCFL(l)\subsetneq CCFL(l+1)$ for all $l\geq 1$.
(3) $CCFL(l)\subsetneq CSL$ for all $l\geq 1$.

Sometimes, it is useful to "neglect" the relations between the components of a parenthesis defined by any $G = (\mathcal{K}, T, P, S) \in CCFG$ for a short time. Then, we investigate instead of G the grammar $G' := (comp(\mathcal{K}), T, P', S)$ for

$$P' := \bigcup_{(k_1,\ldots,k_r)\to(\alpha_1,\ldots,\alpha_r)\in P} \{k_i \to \alpha_i \mid 1 \leq i \leq r\}.$$

Since G' is certainly a context-free grammar, we denote G' (resp. P') by $CF(G)$ (resp. $CF(P)$) in the sequel. Obviously, G' satisfies $L(G) \subseteq L(G')$.

Note: Throughout this paper, we assume w.l.o.g. that $\varepsilon \notin L(G)$ for any $G \in CFG$ or $G \in CCFG$. This simplifies the presentation while everything presented could also be done if $\varepsilon \in L(G)$ holds.

3. The Context-Free Case

In this section, we first introduce the understanding of production sets as systems of formal equations. In [8], this and the Greibach normalform for context-free grammars are used to prove very elegantly a variant of the representation theorem of Shamir. Here, we present the idea of this constructive proof since it allows to translate the word problem for context-free languages into the search for a path in a directed graph. Its edges are marked such that the input word is in the language analysed if and only if the path we search is marked by a sequence of parentheses correctly nested. We define this graph before closing by the algorithm itself.

3.1. The Theorem of Shamir

Let $G = (N, T, P, S)$ a context-free grammar in Greibach normalform, i.e. $P \subseteq N \times TN^*$. Instead of considering each $p \in P$ as a substitution rule, we can take it as a formal equation. Here, the elements of N represent the variables while terminal symbols are treated as constants. Thus, $X \to YbCD \in P$, e.g., results in $X = YbCD$. We can manipulate these equations as usual in the algebra, whereby the multiplication of the variables is associative, but not commutative. As done in [3], we can assign to each P an equation system for formal power series. This system is solvable by algebraic series. In general, we can use this understanding to prove results in the theory of context-free languages by elegant algebraic means. The well-known representation theorems of Chomsky–Schützenberger in [3] and of Shamir in [19] are developed on this background. In addition, this algebraic view can be used to construct the Greibach normalform for any given context-free grammar causing only a cubic growth of the grammar size (cf. [18]).

Now, let $G = (N, T, P, S)$ in Greibach normalform. Then, all equations resulting from P are of the form

$$X = tX_1X_2\ldots X_k, \quad X, X_1, \ldots, X_k \in N,\ t \in T.$$

Therefrom, we can uniquely define the mapping

$$\varphi_G(t) := \left\{X^{-1}u^R \mid X \to tu \in P\right\}$$

for each $t \in T$, where u^R is defined as $(w_1w_2\ldots w_k)^R := w_k\ldots w_2w_1$, while X_i^{-1} is the formal inverse of the variable $X_i \in N$. Intuitively, X^{-1} means "*replace* X", while X says "*generate* X". Therefore, each element of $\varphi_G(t)$ represents one possibility how

to generate $t \in T$ during a derivation process relative to G. For any context-free grammar G, φ_G just defined can be extended to a monoid homomorphism

$$\varphi_G : T^* \longrightarrow \wp\left((N \cup N^{-1})^*\right)$$

where the concatenation is the only monoid operation. $\wp\left((N \cup N^{-1})^*\right)$ denotes the set of all finite subsets of $(N \cup N^{-1})^*$. In [8], φ_G is used to prove the following variant of the representation theorem of Shamir in [19] in an elegant and constructive way:

Theorem 3.1. For any context-free language $L \subseteq T^+$, there exists an alphabet N, a distinguished $S \in N$, and a monoid homomorphism $\varphi_G : T^* \longrightarrow \wp\left((N \cup N^{-1})^*\right)$ such that it holds with $D(N) := ED(\{(X, X^{-1}) \mid X \in N\}, \emptyset)$:

$$w \in L \iff \varphi_G(w) \cap S^{-1} \cdot D(N) \neq \emptyset$$

Besides, it holds $\varphi_G(t) \subset N^{-1}N^*$ for all $t \in T$.

This theorem and φ_G could be used to solve the word problem for $w \in T^+$ relative to G as follows:

1. For each $t \in T$, regard $\varphi_G(t)$ as a formal sum instead of a set of products.
2. Replace each $w_i \in T$, $1 \leq i \leq n$, in $w = w_1 \ldots w_n$ by the formal sum $\varphi_G(w_i)$.
3. Multiply the resulting product of sums out. During this process, substitute subsequences $X_iX_i^{-1}$, $X_i \in N$, by ε. I.e., reduce relative to the congruence relation $\rho_G := \{(X_iX_i^{-1}, \varepsilon) \mid X_i \in N\}$.
4. From Theorem 3.1., it follows that the result contains S^{-1} if and only if $w \in L(G)$ holds.

The representation as product of sums is very compact. This compactness is destroyed when multiplying out. In the worst case, we get $O(|P|^n)$ partial sums which have to be tested modulo ρ_G. Thus, when realizing the above idea, we would get an exponential runtime. Consequently, we have to work on the original representation to get an efficient procedure. Here, we do that by translating the product of sums to a graph and then working on this graph. This answers the word problem without enumerating all possible derivations.

3.2. The Greibach Graph

The main idea is to construct a *column* of a graph for each w_i in $w = w_1 \ldots w_n$. The sequence of these columns equals the sequence of the symbols in w.

At first, the graph gets a start node representing column 0 (it also represents the only row in column 0). $|\varphi_G(w_1)|$ edges directed to column 1 start there.

Each of the columns 1 up to n consists of $|\varphi_G(w_i)|$ rows if we construct the column for w_i. Each *row* represents an element of $\varphi_G(w_i)$ and contains as many nodes as the element it represents contains nonterminals. Let this length be denoted by $|r_j^{w_i}|$. From column 1 to n, there are two kinds of edges: inside a row and from column to column. *Inside* a row j, there are only edges from node v to $v+1$, $1 \leq v < |r_j^{w_i}|$. The l-th edge in this sequence is marked by the $(l+1)$-th symbol of the element of $\varphi_G(w_i)$ represented by row j. *Between columns*, there are only edges from column i to column $i+1$, $0 \leq i < n$. For each row in column i, there are $|\varphi_G(w_i)|$ edges starting at the last node in the row. They are directed to all the different starting nodes of the rows in column $(i+1)$. All of these edges leading to the same row in column $(i+1)$ are marked by the same symbol, namely the first symbol of the element of $\varphi_G(w_{i+1})$ represented by this row. Figure 1 shows an example of such a row.

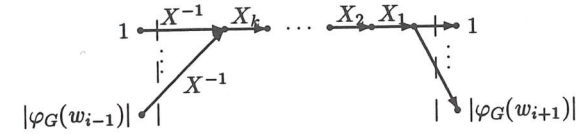


Figure 1 A row in column i of the graph resulting from $X^{-1}X_k \dots X_1 \in \varphi_G(w_i)$

Additionally, the graph gets a final node $\mathcal{F}$ representing column $n+1$. It is the endpoint of $|\varphi_G(w_n)|$ edges marked by ε. These edges start at the different end nodes of the rows in column n.

The connection to the multiplication idea above is that obviously, each term in the result of the formal multiplication corresponds to the marking of a path from the start to the end node in the graph just constructed. Thus, we can say that $w \in L(G)$ holds if and only if in the graph, there is a path whose marking modulo ρ_G equals S^{-1}. The procedure presented in the next section solves exactly this decision problem. To get an elegant description of it, we add a begin node $\mathcal{B}$ to the graph (column -1), which is the starting point of one edge leading to the only node in column 0 and marked by S. Thus, our problem is reduced to a search for a path from $\mathcal{B}$ to $\mathcal{F}$, whose marking modulo ρ_G is ε.

The above construction can be formalized as

Definition 3.1. (Greibach Graph)
Let $G = (N, T, P, S)$ a context-free grammar in Greibach normalform, $w \in T^+$, $n := |w| > 0$. For all $t \in T$, we define $P(t) := \{X \to X_1 \dots X_k \mid X \to tX_1 \dots X_k \in P\}$. The elements of $P(t)$ are enumerated as p_j^t, $1 \le j \le |P(t)|$. GNF-Graph$_G(w)$, the Greibach Graph of w relative to G, *is defined by the node set V_{GNF} and the edge set E_{GNF} given as*

$$V_{GNF} := \{\mathcal{B}\} \cup \{(0,1,0)\} \cup \bigcup_{i=1}^{n} \bigcup_{j=1}^{|P(w_i)|} \bigcup_{l=0}^{|\mathcal{D}(p_j^{w_i})|} \{(i,j,l)\} \cup \{\mathcal{F}\}$$

$$E_{GNF} := \{(\mathcal{B},(0,1,0))\} \cup \{(0,1,0)\} \times \bigcup_{j=1}^{|P(w_1)|} \{(1,j,0)\} \cup$$

$$\bigcup_{i=2}^{n} \Big(\bigcup_{j=1}^{|P(w_{i-1})|} \big\{(i-1,j,|\mathcal{D}(p_j^{w_{i-1}})|)\big\} \times \bigcup_{j=1}^{|P(w_i)|} \{(i,j,0)\} \Big) \cup$$

$$\bigcup_{i=1}^{n} \bigcup_{j=1}^{|P(w_i)|} \bigcup_{l=0}^{|\mathcal{D}(p_j^{w_i})|-1} \{((i,j,l),(i,j,l+1))\} \cup \bigcup_{j=1}^{|P(w_n)|} \big\{(n,j,|\mathcal{D}(p_j^{w_n})|)\big\} \times \{\mathcal{F}\}.$$

The mapping $\gamma : E_{GNF} \longrightarrow N \cup N^{-1} \cup \{\varepsilon\}$, the marking on the edges, is defined as

$$\gamma(e) := \begin{cases} \mathcal{S}(p_j^t)^{-1} & \text{if } t = w_i,\ \mathcal{D}(e) = (i,j,s),\ s = 0. \\ (\mathcal{D}(p_j^t))_s & \text{if } t = w_i,\ \mathcal{D}(e) = (i,j,s),\ s > 0. \\ S & \text{if } \mathcal{S}(e) = \mathcal{B}. \\ \varepsilon & \text{if } \mathcal{D}(e) = \mathcal{F}. \end{cases}$$

γ can be extended to a monoid homomorphism $\gamma : E_{GNF}^* \longrightarrow (N \cup N^{-1})^*$ on the paths in the graph.

Definition 3.2. *Given* $G = (N, T, P, S) \in CFG$, *we define the* parentheses congruence relation for G *by* $\rho_G := \{(X_i X_i^{-1}, \varepsilon) \mid X_i \in N\}$.

Now, we directly get from Theorem 3.1. (cf. [15] for details)

Lemma 3.1. Let $G = (N, T, P, S)$ a context-free grammar in Greibach normalform and $w \in T^+$. It holds:

$$w \in L(G) \iff \text{In } \mathit{GNF\text{-}Graph}_G(w), \text{ there exists a path } u \in E^*_{GNF} \text{ fulfilling } \mathcal{S}(u) = \mathcal{B}, \mathcal{D}(u) = \mathcal{F}, \text{ and } \gamma(u) \equiv_{\rho_G} \varepsilon.$$

Now, we present the algorithm solving the word problem by searching such a path. This is done efficiently by representing partial expressions which are congruent to ε modulo ρ_G by a single edge marked by ε.

3.3. The Analysis

Here, the idea is to search locally for short cuts on all paths from $\mathcal{B}$ to $\mathcal{F}$. These short cuts are represented by a new edge s^* leading from the start to the endpoint of the short cut and being marked by ε. The only thing we do now is trying to prolong these short cuts.

At the start, a short cut corresponds to sequences of two edges with inverse marks. Later on, short cuts are either sequences of two new edges marked by ε (condition $(E4)$ of Definition 2) or sequences of an old edge, a new edge, and finally an old edge where the marks of the two old edges are inverse (condition $(E3)$). Thus, we use the definition of Dyck-sets to develop an efficient procedure. Obviously, our investigation can work locally and treat each new edge only once for the two cases. Each new edge represents compactly a reducible partial expression in the product. No edge is generated twice. We get:

procedure *GNF-analysis*($\mathit{GNF\text{-}Graph}_G(w)$)
begin
$V_* := V_{GNF}$; $E_* := E_{GNF}$; $\mathcal{E} := \emptyset$;
(1) **for all** $s, s' \in E_{GNF}$ **where** $\mathcal{D}(s) = \mathcal{S}(s')$ **and** $\gamma(s), \gamma(s') \neq \varepsilon$
do
if $\gamma(s) \cdot \gamma(s') \equiv_\rho \varepsilon$
then
$s^* := (\mathcal{S}(s), \mathcal{D}(s'))$; $\gamma(s^*) := \varepsilon$;
$E_* := E_* \cup \{s^*\}$; $\mathcal{E} := \mathcal{E} \cup \{s^*\}$;
od;

while $\mathcal{E} \neq \emptyset$
do /* let $s \in \mathcal{E}$ fixed */
(2) **for all** $s' \in E_*$ **where** $\gamma(s') = \varepsilon$ **and** ($\mathcal{D}(s') = \mathcal{S}(s)$ or $\mathcal{S}(s') = \mathcal{D}(s)$)
do
if $\mathcal{D}(s') = \mathcal{S}(s)$
then $s^* := (\mathcal{S}(s'), \mathcal{D}(s))$
else $s^* := (\mathcal{S}(s), \mathcal{D}(s'))$;
$\gamma(s^*) := \varepsilon$;
if $s^* \notin E_*$
then
$E_* := E_* \cup \{s^*\}$; $\mathcal{E} := \mathcal{E} \cup \{s^*\}$;
od;

(3) **for all** $s', s'' \in E_*$ **where** $\gamma(s'), \gamma(s'') \neq \varepsilon$ **and** $\mathcal{D}(s') = \mathcal{S}(s)$ **and** $\mathcal{S}(s'') = \mathcal{D}(s)$

do

if $\gamma(s') \cdot \gamma(s'') \equiv_{\rho G} \varepsilon$

then

$s^* := (\mathcal{S}(s'), \mathcal{D}(s''))$; $\gamma(s^*) := \varepsilon$;

if $s^* \notin E_*$

then

$E_* := E_* \cup \{s^*\}$; $\mathcal{E} := \mathcal{E} \cup \{s^*\}$;

od;

$\mathcal{E} := \mathcal{E} \setminus \{s\}$;

od;

end;

We denote by *GNF–Graph*$^*_G(w)$ the graph (V_*, E_*) determined by *GNF–analysis*(*GNF-Graph*$_G(w)$).

Lemma 3.2. Let $G = (N, T, P, S)$ a context-free grammar in Greibach normalform and $w \in T^+$. It holds:

In *GNF–Graph*$_G(w)$, there exists a path $u \in E^*_{GNF}$ where $\mathcal{S}(u) = \mathcal{B}, \mathcal{D}(u) = \mathcal{F}, \gamma(u) \equiv_{\rho G} \varepsilon$. $\iff$

In *GNF–Graph*$^*_G(w)$, there exists an edge $s^* = (\mathcal{B}, \mathcal{F})$ where $\gamma(s^*) = \varepsilon$.

Proof:

$\Longrightarrow$: We prove: In *GNF–Graph*$_G(w)$, there exists a path $u \in E^*_{GNF}$ where $\mathcal{S}(u) = v$, $\mathcal{D}(u) = v'$, and $\gamma(u) \equiv_{\rho G} \varepsilon$. $\Longrightarrow$
In *GNF–Graph*$^*_G(w)$, there exists $s^* = (v, v') \in E_*$ where $\gamma(s^*) = \varepsilon$.

The proof is done by induction on the length of partial paths $|u|$.

$|u| = 1$: $u \in E_{GNF}$ holds for $\gamma(u) = \varepsilon$.

$|u| = 2$: It exists some $v'' \in V_{GNF}$ where $(v, v''), (v'', v') \in E_{GNF}$ and $\gamma((v, v'')) \cdot \gamma((v'', v')) \equiv_{\rho G} \varepsilon$

$\overset{(1)}{\Longrightarrow} \exists s^* := (v, v') \in E_{GNF} : \gamma(s^*) = \varepsilon$.

$|u| > 2$: CASE 1: $u = u_1 u_2$ where $|u_1|, |u_2| < |u|$, and $\gamma(u_1) \equiv_{\rho G} \varepsilon$, $\gamma(u_2) \equiv_{\rho G} \varepsilon$

$\overset{\text{Ind.}}{\Longrightarrow} \exists s_1 = (\mathcal{S}(u_1), \mathcal{D}(u_1)), s_2 = (\mathcal{S}(u_2), \mathcal{D}(u_2)) \in E_{GNF} :$
$\gamma(s_1) = \gamma(s_2) = \varepsilon$

$\overset{(2)}{\Longrightarrow} \exists s^* := (\mathcal{S}(u_1), \mathcal{D}(u_2)) = (v, v') \in E_{GNF} : \gamma(s^*) = \varepsilon$

CASE 2: $u = k_1 u' k_2$ where $|u'| = |u| - 2$ and $\gamma(u') \equiv_{\rho G} \varepsilon$, and $\gamma(k_1) \cdot \gamma(k_2) \equiv_{\rho G} \varepsilon$

$\overset{\text{Ind.}}{\Longrightarrow} \exists s = (\mathcal{S}(u'), \mathcal{D}(u')) \in E_{GNF} : \gamma(s) = \varepsilon$.

$\overset{(3)}{\Longrightarrow} \exists s^* := (v, v') \in E_{GNF} : \gamma(s^*) = \varepsilon$

Therefrom, part one of Lemma 3.2. follows with $v = \mathcal{B}$ and $v' = \mathcal{F}$.

$\Longleftarrow$: analogous

The correctness of the procedure follows. We get:

Theorem 3.2. Let $G = (N, T, P, S)$ a context-free grammar in Greibach normalform. The procedure *GNF–analysis* solves the word problem for any $w \in T^+$, $n := |w|$, relative to G in time $O(n^3)$ (more exactly $O(|P|^4 \cdot n^2 + |P|^3 \cdot n^3)$).

Proof:
(1) is executed only once. Because of the graph structure, two edges with inverse marking can only follow each other when going on from one column to the next one. Thus, (1) costs time $O(n)$.

(2) and (3) are executed once for each element of $\mathcal{E}$. Since no edge in $\mathcal{E}$ is produced twice and since each edge in $\mathcal{E}$ connects two different columns, $\mathcal{E}$ receives at most $O(n^2)$ elements during the analysis.

(2) can only generate a new ε-edge for a fixed $s \in \mathcal{E}$ if there is another ε-edge s' which shares an endpoint with s. Because of the construction of $GNF\text{–}Graph_G(w)$, there are at most $O(n)$ such edges.

In contrast to this, for any fixed $s \in \mathcal{E}$, (3) demands the existence of two edges s', s'' fulfilling $\gamma(s'), \gamma(s'') \neq \varepsilon$ to produce a new ε-edge. In addition, s' and s'' have to share an endpoint with s. In $GNF\text{–}Graph_G(w)$, there are at most $|P|^2$ such edges.

Remark: Given an acyclic graph with one begin and one final node as well as a marking γ from $\Sigma \cup \Sigma^{-1}$, Σ finite alphabet, and a congruence relation ρ, our algorithm answers the question whether there exists a path u from $\mathcal{B}$ to $\mathcal{F}$ fulfilling $\gamma(u) \equiv_\rho \varepsilon$, too. There, it needs for $O(n)$ nodes $O(n^4)$ operations.

4. The Idea

In the context-free case, it was successful to reduce the wordproblem to the analysis of sequences of parentheses. Therefore, we want to use the same approach for $CCFG(2)$. Consequently, we have to construct a graph where we search for a path, whose marking expresses derivability/non-derivability of the input word relative to the underlying grammar. Here, context-free derivability does not suffice, but we additionally have to test the correctness of the coupling relative to the two components of nonterminal parentheses inside any derivation. Thus, we need a marking on the edges consisting in two different components, one for each of the two different tests to be performed. Now, the idea is to define a Greibach normalform for all $G \in CCFG(2)$ such that the graph and the first component of the marking (modelling context-free derivability) can be constructed as before using $CF(G)$ instead of G.

To get the second component of the marking, each $(X, \overline{X}) \to (\alpha, \overline{\alpha}) \in P$ is uniquely named by $(p, \overline{p})$, $p \in \Sigma$, Σ arbitrary alphabet. $(p, \overline{p})$ is understood as a parenthesis, i.e. we define the parentheses set $\pi_\Sigma := \{(p, \overline{p}) \mid p \in \Sigma, \nu^{-1}(p) \in P\}$. Thus, we can define a new congruence relation as $\sigma_G := \{(p\overline{p}, \varepsilon) \mid (p, \overline{p}) \in \pi_\Sigma\}$. Now, it remains to use p resp. $\overline{p}$ as second marking such that it holds:

$$w \in L(G) \Longleftrightarrow \exists u \in E^*_{GNF} : \mathcal{S}(u) = \mathcal{B},\ \mathcal{D}(u) = \mathcal{F},\ \gamma_1(u) \equiv_{\rho_G} \varepsilon,\ \gamma_2(u) \equiv_{\sigma_G} \varepsilon$$

We do that by taking p resp. $\overline{p}$ as second component for edges marked in the first component by X^{-1} resp. $\overline{X}^{-1}$, where X^{-1} resp. $\overline{X}^{-1}$ resulted from those terms in φ_G coming from $X \to \alpha$ resp. $\overline{X} \to \overline{\alpha}$ (i.e. the left-hand side of the productions). All other second components of the marking are set to ε. The whole idea is made more concrete in the sequel.

4.1. The Normalform

Let $G \in CCFG(2)$. To generalize the algorithm, we first have to generalize the context-free Greibach normalform. Therefore, it seems to be the best to demand Greibach normalform for $CF(G)$. Obviously, this generalization could easily be extended for $CCFG(l)$, $l > 2$. It leads to the following situation:

Example 4.1. Let $G := (\{S,(A,\overline{A}),(B,\overline{B}),(X,\overline{X})\},\{a,b,c,\$\},P,S) \in CCFG(2)$ where

$$P := \{S \to \$X\overline{X},\ (X,\overline{X}) \overset{(f,\overline{f})}{\to} (aBC,a\overline{CB}),(B,\overline{B}) \overset{(g,\overline{g})}{\to} (b,c),\ (C,\overline{C}) \overset{(h,\overline{h})}{\to} (c,b)\}.$$

G shows the normalform demanded and it holds $L(G) = \{\$abcabc\}$.

To simplify the presentation at this point, we use the product of sums (cf. the consequences of Theorem 3.2.) for the analysis instead of the Greibach Graph. The two components of the marking are modelled by two separate products written in parallel. Now, the analysis of $w := \$abcabc \in L(G)$ results in:

$$\begin{array}{ccccccccccccc} \$ & \cdot & a & \cdot & b & \cdot & c & \cdot & a & \cdot & b & \cdot & c \\ \left(S^{-1}\overline{X}X\right)\cdot\begin{pmatrix}X^{-1}CB\\ +\overline{X}^{-1}\overline{B}\,\overline{C}\end{pmatrix}\cdot\begin{pmatrix}B^{-1}\\ +\overline{C}^{-1}\end{pmatrix}\cdot\begin{pmatrix}\overline{B}^{-1}\\ +C^{-1}\end{pmatrix}\cdot\begin{pmatrix}X^{-1}CB\\ +\overline{X}^{-1}\overline{B}\,\overline{C}\end{pmatrix}\cdot\begin{pmatrix}B^{-1}\\ +\overline{C}^{-1}\end{pmatrix}\cdot\begin{pmatrix}\overline{B}^{-1}\\ +C^{-1}\end{pmatrix} \\ \varepsilon \quad \cdot \quad \begin{pmatrix}f\\ +\overline{f}\end{pmatrix} \quad \cdot \begin{pmatrix}g\\ +\overline{h}\end{pmatrix} \cdot \begin{pmatrix}\overline{g}\\ +h\end{pmatrix} \cdot \quad \begin{pmatrix}f\\ +\overline{f}\end{pmatrix} \quad \cdot \begin{pmatrix}g\\ +\overline{h}\end{pmatrix} \cdot \begin{pmatrix}\overline{g}\\ +h\end{pmatrix} \end{array}$$

This means that a correct derivation via the expression

$$S^{-1}\overline{X}XX^{-1}CBB^{-1}C^{-1}\overline{X}^{-1}\overline{B}\,\overline{C}\,\overline{C}^{-1}\overline{B}^{-1} \equiv_{\rho_G} S^{-1}$$

corresponds to the following sequence in the second product:

$$\varepsilon \cdot fgh\overline{f}\,\overline{h}\,\overline{g}$$

This is *not* an element of a Semi-Dyck set. This cannot be reduced to ε using ρ_G.

The above example signifies that such a generalization is not suitable for our purpose. Thus, we need a more sophisticated normalform to perform the context-free as well as the coupled part of the analysis by reducing relative to a congruence relation.

When constructing φ_G, each $X \to tX_1 \ldots X_k$ is transformed into $t = X^{-1}X_k \ldots X_1$. Inside the graph, $X_1^{-1}, \ldots, X_k^{-1}$ appear on the right of this sequence. In addition, the nonterminal components coupled to $X_1, \ldots, X_k$ have to appear still farther on the right. To produce such a correct sequence of parentheses, $\overline{X}^{-1}$ has to be situated on the right of these coupled components. This can be achieved, if the second component of coupled productions is of a symmetric form as the first one is. We get:

Definition 4.1. (Generalized GNF)
$G = (\mathcal{K},T,P,S) \in CCFG(2)$ *is in* Generalized Greibach Normalform (GNF), *if and only if it satisfies:*

(1) $P|_{\mathcal{K}[1]} \subseteq \mathcal{K}[1] \times T \cdot (comp(\mathcal{K}))^*$.

(2) $P|_{\mathcal{K}[2]} \subseteq \mathcal{K}[2] \times (T \cdot (comp(\mathcal{K}))^* \times (comp(\mathcal{K}))^* \cdot T)$.

Remark: Up to now, it is still open how to construct a $G' \in CCFG(2)$ in generalized Greibach normalform fulfilling $L(G) = L(G')$ for each $G \in CCFG(2)$.

Let $G = (\mathcal{K},T,P,S) \in CCFG(2)$ in generalized GNF. Using $CF(P)$, we can define φ_G uniquely in analogy to the context-free case. The symmetric form of the productions leads to

$$\varphi_G(t) := \left\{X^{-1}u^R \mid X \to tu \in CF(P)\right\} \cup \left\{u^R\overline{X}^{-1} \mid \overline{X} \to ut \in CF(P)\right\}.$$

This can also be extended to a monoid homomorphism

$$\varphi_G : T^* \longrightarrow \wp\left((comp(\mathcal{K}) \cup comp(\mathcal{K})^{-1})^*\right).$$

This homomorphism defines the first component of the marking of the edges for our graph as usual.

Using this definition, our example looks as follows:

Example 4.2. (Example 4.1. continued)
Now, let $G := (\{S, (A, \overline{A}), (B, \overline{B}), (X, \overline{X})\}, \{a, b, c, \$\}, P, S)$ be defined with

$$P := \{S \to \$X\overline{X},\ (X, \overline{X}) \overset{(f,\overline{f})}{\to} (aBC, \overline{CB}c),\ (B, \overline{B}) \overset{(g,\overline{g})}{\to} (b, b),\ (C, \overline{C}) \overset{(h,\overline{h})}{\to} (c, a)\}.$$

This G shows generalized Greibach normalform, and we still have $L(G) = \{\$abcabc\}$. Using the product representation as before, the analysis of $w := \$abcabc \in L(G)$ now results in:

$$\begin{array}{ccccccccccccc} \$ & \cdot & a & \cdot & b & \cdot & c & \cdot & a & \cdot & b & \cdot & c \\ \left(S^{-1}\overline{X}X\right) & \cdot & \begin{pmatrix} X^{-1}CB \\ +\overline{C}^{-1} \end{pmatrix} & \cdot & \begin{pmatrix} B^{-1} \\ +\overline{B}^{-1} \end{pmatrix} & \cdot & \begin{pmatrix} \overline{B}\,\overline{C}\,\overline{X}^{-1} \\ +C^{-1} \end{pmatrix} & \cdot & \begin{pmatrix} X^{-1}CB \\ +\overline{C}^{-1} \end{pmatrix} & \cdot & \begin{pmatrix} B^{-1} \\ +\overline{B}^{-1} \end{pmatrix} & \cdot & \begin{pmatrix} \overline{B}\,\overline{C}\,\overline{X}^{-1} \\ +C^{-1} \end{pmatrix} \\ \varepsilon & \cdot & \begin{pmatrix} f \\ +\overline{h} \end{pmatrix} & \cdot & \begin{pmatrix} g \\ +\overline{g} \end{pmatrix} & \cdot & \begin{pmatrix} \overline{f} \\ +h \end{pmatrix} & \cdot & \begin{pmatrix} f \\ +\overline{h} \end{pmatrix} & \cdot & \begin{pmatrix} g \\ +\overline{g} \end{pmatrix} & \cdot & \begin{pmatrix} \overline{f} \\ +h \end{pmatrix} \end{array}$$

Obviously, the sequence $\varepsilon \cdot fgh\overline{h}\overline{g}\overline{f}$ correctly nested relative to Π_Σ is contained in the result set of the second product. Our problem is solved.

Notation 2 *To simplify the presentation, we will talk about* red *and* black *edges in the sequel. Thereby, edges marked by X^{-1} or $\overline{X}^{-1}$ (the former $\mathcal{S}(p)$ for some $p \in CF(P)$) are said to be red, all the others black. In the figures, red edges are represented by thick lines while black edges are represented dotted.*

4.2. The Modified Graph

Let $G = (\mathcal{K}, T, P, S) \in CCFG(2)$ in generalized GNF. For any $w \in T^*$, the edges in $GNF\text{-}Graph_{CF(G)}(w)$ are constructed as before using $CF(G)$ since we need a context-free grammar. The differences consist in the marking of the edges. They are:

1. Each edge $s \in E_{GNF}$ is marked by $\gamma(s) := (\gamma_1(s), \gamma_2(s))$. Here, γ_1 results from $\varphi_G(t)$ as in the context-free case. For $\gamma_2(s)$, the following holds:

   $\gamma_2(s) = \varepsilon$, if s is a black edge, i.e. $\gamma_1(s) \in comp(\mathcal{K})$, or if $\gamma_1(s) = X^{-1} \in \mathcal{K}[1]^{-1}$.

   $\gamma_2(s) := \nu(p)$, if s is a red edge and $\gamma_1(s) = X^{-1}$ resulted from $X \to tX_1 \dots X_k \in CF(P)$, the first component of $p := (X, \overline{X}) \to (tX_1 \dots X_k, \alpha) \in P$.

   $\gamma_2(k) := \overline{\nu(q)}$, if s is a red edge and $\gamma_1(s) = \overline{X}^{-1}$ resulted from $X \to X_1 \dots X_k t \in CF(P)$, the second component of $q := (Y, X) \to (\alpha, X_1 \dots X_k t) \in P$.
2. Now, the elements of $\varphi_G(t)$ are of the form

   $$X^{-1}X_k \dots X_1 \in \varphi_G(t) \qquad (a)$$

   for first components of productions (here, $X \to tX_1 \dots X_k \in CF(P)$ holds), while second components ($\overline{X} \to X_1' \dots X_l' t' \in CF(P)$) they are as

   $$X_l' \dots X_1' \overline{X}^{-1} \in \varphi_G(t'). \qquad (b)$$

   Graphically, the difference is shown for $p := \nu((X, \overline{X}) \to (tX_1 \dots X_k, \alpha))$, $q := \nu((X, \overline{X}) \to (\beta, X_1' \dots X_l' t'))$ in Figure 2. We see that red edges appear on the left of their respective black edges except from the situation if their (red) marking resulted from the second component of a coupled production.
3. The additional edge $(\mathcal{B}, (0, 1, 0))$ necessary to have a uniform context-free analysis is not needed here. Thus, we remove it and identify $\mathcal{B}$ and $(0, 1, 0)$.

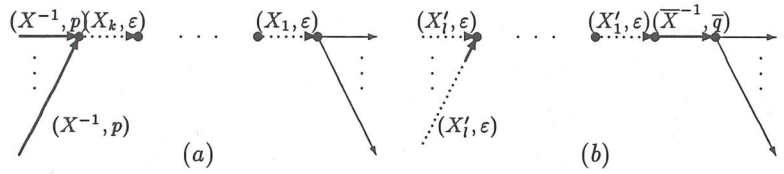


Figure 2 $\gamma := (\gamma_1, \gamma_2)$, the marking of the edges

4. The graph resulting from $G \in CCFG(2)$ in generalized GNF via $CF(G)$ in this way is denoted by $NF\text{-}Graph_G(w)$.

We finish the graph construction by an example also used to explain the algorithm.

Example 4.3. Let $G = (\mathcal{K}, T, P, S) \in CCFG(2)$ where $\mathcal{K} := \{S, (X, \overline{X}), (Y, \overline{Y})\}$, $T := \{\$, a, b, c, d\}$ and $P := \{S \to \$X\overline{X},\ (Y,\overline{Y}) \overset{(h,\overline{h})}{\to} (b,c),\ (X,\overline{X}) \overset{(f,\overline{f})}{\to} (aXY, \overline{Y}\,\overline{X}d),\ (X,\overline{X}) \overset{(g,\overline{g})}{\to} (aY, \overline{Y}d)\}$. Thus, we have $L(G) = \{\$a^n b^n c^n d^n \mid n \geq 1\}$. P defines the mapping φ_G for all $t \in T$ as

$$\begin{aligned}
\varphi(\$) &= S^{-1}\overline{X}X,\\
\varphi(a) &= X^{-1}YX + X^{-1}Y,\\
\varphi(b) &= Y^{-1},\\
\varphi(c) &= \overline{Y}^{-1},\\
\varphi(d) &= \overline{X}\,\overline{Y}\,\overline{X}^{-1} + \overline{Y}\,\overline{X}^{-1}.
\end{aligned}$$

For $w := \$aabbccdd$, $NF\text{-}Graph_G(w)$ is shown in Figure 4 while its derivation tree is shown in Figure 3. To simplify Figure 4, we omit $\gamma_2(s)$ for all edges s where $\gamma_2(s) = \varepsilon$. For all the others, γ_1 and γ_2 are written separately on different sides of the edge.

4.3. The Principle of the Analysis

Let $G = (\mathcal{K}, T, P, S) \in CCFG(2)$ in generalized GNF.

As we already scetched, the idea is to search a path u in $NF\text{-}Graph_G(w)$, which shows a sequence of parentheses correctly nested in the first as well as in the second component of its marking. We still have to show how this path can be found efficiently.

The first idea is to use the context-free algorithm on γ_1. Each time a new ε-edge is created γ_2 could be collected from the edges used for this construction. This means that we could not forget the partial paths leading to the construction of the ε-edge as it

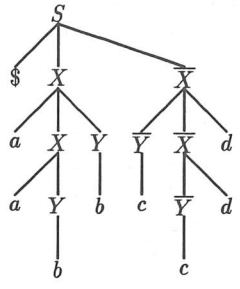


Figure 3 Derivation tree for $w := \$aabbccdd$

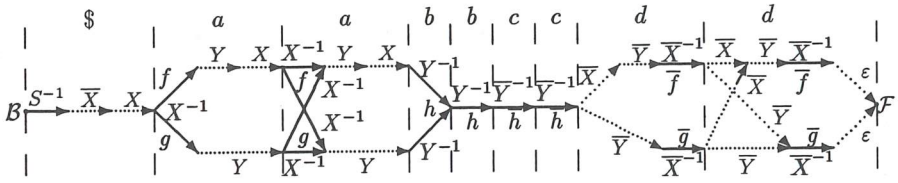


Figure 4 *NF-Graph*$_G(w)$ for $w = \$aabbccdd$

was possible before. Since the context-free and the coupled information are distributed differently on the graph and since in general, they cannot be reduced at the same points, the consequence would be that each ε-edge could show an exponential amount of second-component markings. This is too expensive for an efficient procedure. Besides, since this procedure would not take special properties of the graph into account, an efficient solution would mean the efficient solution of an NP-complete problem as shown in the Appendix.

Thus, we decide to trace the derivation process for X and $\overline{X}$ in $(X, \overline{X}) \in \mathcal{K}[2]$ in parallel. This means to search at first for couplings. Therefore, it suffices to examine red edges and the second component of their marking. Relative to the couplings found we test context-free derivability. More concrete, this means to investigate the context-free derivability of the black edges belonging to these red edges, i.e. to investigate the first component of their marking. Both operations are performed in turn.

We start with the most simplest cases, which are investigated in a preprocessing performed only once. These cases stand for $N \to w_i \in P$, $N \in \mathcal{K}[1]$, and $(X, \overline{X}) \to (w_j, w_{j+1}) \in P$, $(X, \overline{X}) \in \mathcal{K}[2]$. They are easy to find since they are represented by one ore two continuous red edges only. They model the smallest correct sequences of parentheses (ε or $p\overline{p}$) we can find in the whole graph.

Starting from here, the procedure incrementally constructs continuously growing paths in *NF-Graph*$_G(w)$. The first condition each path u has to fulfill is $\gamma_2(u) \equiv_{\rho_G} \varepsilon$. Therefore, we only test red edges. Following the definition of $ED(\mathcal{K}, T)$, there are 3 cases to distinguish:

1. $u, v \in ED(\pi_\Sigma, \emptyset) \implies u \cdot v \in ED(\pi_\Sigma, \emptyset)$
2. $N \in \mathcal{K}[1], v \in ED(\pi_\Sigma, \emptyset) \implies \varepsilon_N \cdot v \in ED(\pi_\Sigma, \emptyset)$
3. $(p, \overline{p}) \in \pi_\Sigma, v \in ED(\pi_\Sigma, \emptyset) \implies pv\overline{p} \in ED(\pi_\Sigma, \emptyset)$

Hereby, 2. is a special case of 3. and models the situation of red edges s, where $\gamma_2(s) = \varepsilon$ holds, i.e. $(\gamma_1(s))^{-1} \in \mathcal{K}[1]$. The algorithm realizes these conditions as 3 separate steps which are iterated successively.

The second condition each path u has to fulfill is, if u leads from column i to j and if $\gamma_1(u) \equiv_{\rho_G} W$, $W^{-1} \overset{*}{\Rightarrow}_G w_i \dots w_j$ has to hold. This condition is tested separately inside each partial step.

Example 4.4. (Example 4.3 continued)
Relative to the derivation tree in Figure 3, the algorithm works bottom-up and from the inside to the outside of the tree. Thus, it finds at first the coupling $(Y, \overline{Y}) \to (b, c)$ (graph columns 5/6) in the preprocessing, again $(Y, \overline{Y}) \to (b, c)$ (columns 4/7) then $(X, \overline{X}) \to (aY, \overline{Y}d)$ (columns 3/8) thereby reducing with σ_G, $(X, \overline{X}) \to (aXY, \overline{Y}\overline{X}d)$ (columns 3/8) - all by Step 3 - and finally $S \to \$X\overline{X}$ by Step 2.

5. The Algorithm

5.1. Basics

Before formalizing the algorithm, we need the following notions to describe it exactly:

Notation 3 *Let $G \in CCFG(2)$ in generalized GNF, $w \in T^+$. The set of all red edges in NF-Graph$_G(w)$ is denoted by $RED_G(w)$.*

Notation 4 *Let $G = (\mathcal{K}, T, P, S) \in CCFG(2)$. As formal calculation rule, we define $(X^{-1})^{-1} := X$ for $X \in comp(\mathcal{K})$. Additionally, for any $X_1, \ldots, X_j \in comp(\mathcal{K})$, it holds $v := X_1^{-1} \cdot \ldots \cdot X_j^{-1} \implies v^{-1} = X_1 \cdot \ldots \cdot X_j$.*

Notation 5 *Let $s \in RED_G(w)$ for $G = (\mathcal{K}, T, P, S) \in CCFG(2)$, $w \in T^+$. We define:*

- *$Col(s) := i \iff \mathcal{D}(s) = (i, j, k)$.*
- *$Prod(s) := \gamma_1(u) = v \in (comp(\mathcal{K}))^+$*

  $\iff \gamma_1(s)^{-1} \to w_{Col(s)} v^R \in CF(P)$ *and* $\mathcal{D}(s) = \mathcal{S}(u)$ *or*

  $\gamma_1(s)^{-1} \to v^R w_{Col(s)} \in CF(P)$ *and* $\mathcal{D}(u) = \mathcal{S}(s)$.

In its course, the algorithm marks the couplings it finds by green edges (the new "ε-edges"). All of them are collected in $GREEN_G(w)$. Green edges generated by Step 1 or Step 3 have to store what is contained inside this coupling. This is done as follows:

Definition 5.1. (single, composed)
Let $G = (\mathcal{K}, T, P, S) \in CCFG(2)$ in generalized GNF, $w \in T^+$. $k \in GREEN_G(w)$ is called single, *if there is no mark on k. Otherwise, k is called* composed. *The marks of a composed k are denoted by $\underline{marks}(k)$. It consists of the lists $\underline{linear}(k)$ determined by Step 1, and $\underline{nested}(k)$ constructed by Step 3.*

The red edges stored by these sequences of green edges with markings can be denoted as follows:

Definition 5.2. (Forrest, Yield)
Let $G = (\mathcal{K}, T, P, S) \in CCFG(2)$ in generalized GNF, $w \in T^+$. For all $k := (s, s') \in GREEN_G(w)$, the directed graph $Forrest(k) := (V(k), E(k))$ and the mapping $Yield(k)$ are defined as:

1. Let $k = (s, s')$ single.
If $(\gamma_1(s))^{-1} \in \mathcal{K}[1]$, we have $Forrest(k) := (\{s, k\}, \{(k, s)\})$, $Yield(k) := \gamma_1(s)$.
If $\left((\gamma_1(s))^{-1}, (\gamma_1(s'))^{-1}\right) \in \mathcal{K}[2]$, we have $Forrest(k) := (\{s, s', k\}, \{(k, s), (k, s')\})$, $Yield(k) := \gamma_1(s) \cdot \gamma_1(s')$.

2. Let $k = (s, s')$ nested and marked by $linear(k)$ consisting of $m_{lin} \geq 0$ marks to green edge pairs (k_1^j, k_2^j), $0 \leq j \leq m_{lin}$, as well as by $nested(k)$ consisting of $m_{nes} \geq 0$ marks to single green edges k^j, $0 \leq j \leq m_{nes}$. It holds $m_{lin} + m_{nes} > 0$. We have

$$Forrest(k) := \bigcup_{j=1}^{m_{lin}} Forrest_{lin}(k, j) \cup \bigcup_{j=1}^{m_{nes}} Forrest_{nes}(k, j),$$

$$Yield(k) := \bigcup_{j=1}^{m_{lin}} Yield_{lin}(k, j) \cup \bigcup_{j=1}^{m_{nes}} Yield_{nes}(k, j).$$

For all $(k_1^j, k_2^j) \in linear(k)$, it holds

$$Forrest_{lin}(k, j) := (V(k_1^j) \cup V(k_2^j) \cup \{k\}, E(k_1^j) \cup E(k_2^j) \cup \{(k, k_1^j), (k, k_2^j)\}),$$
$$Yield_{lin}(k, j) := \{u \cdot v \mid u \in Yield(k_1^j), v \in Yield(k_2^j)\}.$$

For all $k^j \in nested(k)$, we have

$$Forrest_{nes}(k,j) := \left(V(k^j) \cup \{k,s,s'\}, E(k^j) \cup \{(k,s),(k,k^j),(k,s')\}\right),$$
$$Yield_{nes}(k,j) := \{\gamma_1(s) \cdot u \cdot \gamma_1(s') \mid u \in Yield(k^j)\}.$$

The algorithm fulfills as invariants the content of the following two lemmata. They are formally proved in Section 5.3 by induction on the distance of the columns any $k \in GREEN_G(w)$ connects.

Lemma 5.1. Let $G \in CCFG(2)$ in generalized GNF and $w \in T^+$, $n := |w|$. For all $k \in GREEN_G(w)$ where $Col(\mathcal{S}(k)) = i$ and $Col(\mathcal{D}(k)) = j$, it holds:

$$v \in Yield(k) \implies v^{-1} \overset{*}{\Rightarrow}_G w_i \dots w_j$$

Lemma 5.2. Let $G \in CCFG(2)$ in generalized GNF and $w \in T^+$ where $n := |w|$. For all $W \in ED(\mathcal{K}, \emptyset)$, it holds:

$$W \overset{*}{\Rightarrow}_G w_i \dots w_j \implies \exists k \in GREEN_G(w) : Col(\mathcal{S}(k)) = i, Col(\mathcal{D}(k)) = j, \; W^{-1} \in Yield(k)$$

From these lemmata, the correctness of our procedure follows, since then, we have:

$$w \in L(G) \iff \text{Following the algorithm, } \exists k \in GREEN_G(w) \text{ fulfilling } Col(\mathcal{S}(k)) = 1,\ Col(\mathcal{D}(k)) = n, \text{ and } Yield(k) = S^{-1}.$$

5.2. The Procedure

Let $G = (\mathcal{K}, T, P, S) \in CCFG(2)$ in generalized GNF, $w \in T^+$, $n := |w|$, $\rho_G := \{(XX^{-1}, \varepsilon) \mid X \in comp(\mathcal{K})\}$, $\pi_\Sigma := \{(p, \overline{p}) \mid p \in \Sigma\}$, and $NF\text{-}Graph_G(w)$ constructed. The algorithm can be formalized as follows:

PREPROCESSING

$GREEN_G(w) := \emptyset$;
for all $s \in RED_G(w)$ **where** ($Col(s) = i$ **and** $(\gamma_1(s))^{-1} \to w_i \in P$)
do
  $GREEN_G(w) := GREEN_G(w) \cup \{(s,s)\}$;
  $single((s,s)) := true$;
od;
for all $s, s' \in RED_G(w)$ **where** $(\mathcal{D}(s) = \mathcal{S}(s'))$
do
  if $((\gamma_1(s))^{-1}, (\gamma_1(s'))^{-1}) \in \mathcal{K}[2]$ **and** $(\gamma_2(s), \gamma_2(s')) \in \pi_\Sigma$
  then
    $GREEN_G(w) := GREEN_G(w) \cup \{(s,s')\}$;
    $single((s,s')) := true$;
od;

ITERATION

for $d := 1$ **to** $(n+2)$
do
  for all $s, s' \in RED_G(w)$ **where** $Col(s') - Col(s) = d$
  do
(1) **for all** $k, k' \in GREEN_G(w)$ **where** $(\mathcal{S}(k) = s, \mathcal{D}(k') = s'$ **and** $\mathcal{D}(\mathcal{D}(k)) = \mathcal{S}(\mathcal{S}(k'))$)
    do
      $k^* := (s,s')$;

if $k^* \notin GREEN_G(w)$
then
$GREEN_G(w) := GREEN_G(w) \cup \{k^*\}$;
$single(k^*) := false$;
fi;
$linear(k^*) := linear(k^*) \cup \{(k, k')\}$;
od;

(2) **if** $(\gamma_1(s))^{-1} \in \mathcal{K}[1]$
then
if $\exists k \in GREEN_G(w)$ **where** $((Col(\mathcal{S}(k)) = Col(s) + 1$) **and** $(\mathcal{D}(k) = s')$
and $\exists v \in Yield(k)$ **where** $(Prod(s) \cdot v \equiv_{\rho_G} \varepsilon))$
then
$k^* := (s, s')$;
if $k^* \notin GREEN_G(w)$
then $GREEN_G(w) := GREEN_G(w) \cup \{k^*\}$;
$single(k^*) := true$;

(3) **if** $\left((\gamma_1(s))^{-1}, (\gamma_1(s'))^{-1}\right) \in \mathcal{K}[2]$ **and** $(\gamma_2(s), \gamma_2(s')) \in \pi_\Sigma$
then
for all $k \in GREEN_G(w)$ **where** ($Col(\mathcal{S}(k)) = Col(s) + 1$ **and**
$Col(\mathcal{D}(k)) = Col(s') - 1$)
do
$k^* := (s, s')$;
for all $v_1 v_2 v_3 \in Yield(k)$ **where** (a) $Prod(s) \cdot v_1 \equiv_{\rho_G} \varepsilon$
(b) $(v_3 \cdot Prod(s'))^R \equiv_{\rho_G} \varepsilon$
do
if $k^* \notin GREEN_G(w)$
then
$GREEN_G(w) := GREEN_G(w) \cup \{k^*\}$;
$single(k^*) := false$;
if $v_2 = \varepsilon$
then $single(k^*) := true$;
else
Let $k' \in V(k)$ **where** $Yield(k') = v_2$ in $Forrest(k) = (V(k), E(k))$;
$nested(k^*) := nested(k^*) \cup \{k'\}$;
/* $\left((\gamma_1(\mathcal{S}(k^*)))^{-1}, (\gamma_1(\mathcal{D}(k^*)))^{-1}\right) \in \mathcal{K}[2]$ */
od;
od;
fi;
od;
od;

5.3. The Correctness

In the sequel, we prove the two lemmata ensuring the correctness of our algorithm.

Proof (of Lemma 5.1.)
Here, we often use the following simple properties:

(E_1) $Col(s) = i$ for $s \in RED_G(w)$ where $\left((\gamma_1(s))^{-1} \in \mathcal{K}[1]\right)$ or

$\left(\exists \overline{X} \in comp(\mathcal{K}) : \left((\gamma_1(s))^{-1}, \overline{X}\right) \in \mathcal{K}[2]\right)$

$$\Longrightarrow (\gamma_1(s))^{-1} \to w_i(Prod(s))^R \in CF(P)$$

$Col(s) = i$ for $s \in RED_G(w)$ where $(\exists X \in comp(\mathcal{K}) : \left(X, (\gamma_1(s))^{-1}\right) \in \mathcal{K}[2])$

$$\Longrightarrow (\gamma_1(s))^{-1} \to (Prod(s))^R w_i \in CF(P)$$

(E_2) Let $u_1 \in RED_G^*(w)$ and $u_2 \in (E_{GNF} \setminus RED_G(w))^*$. We have:

$$\gamma_1(u_2) \cdot \gamma_1(u_1) \equiv_{\rho_G} \varepsilon \Longrightarrow (\gamma_1(u_1))^{-1} = (\gamma_1(u_2))^R$$

$$(\gamma_1(u_2) \cdot \gamma_1(u_1))^R \equiv_{\rho_G} \varepsilon \Longrightarrow (\gamma_1(u_1))^{-1} = (\gamma_1(u_2))^R$$

The algorithm generates single and composed green edges. For each of them, we need to examine a separate case. Thus, we prove that in each step, the algorithm ensures the following two invariants for all $0 \leq i \leq n$ and all $k \in GREEN_G(w)$ where $Col(\mathcal{S}(k)) = i$ and $\delta(k) := Col(\mathcal{D}(k)) - Col(\mathcal{S}(k)) \leq n$:

$(\mathcal{I}_1)$ k single $\Longrightarrow (Yield(k))^{-1} \overset{*}{\Rightarrow}_G w_i \ldots w_{i+\delta(k)}$

$(\mathcal{I}_2)$ k composed $\Longrightarrow \forall v \in Yield(k)$: $v^{-1} \overset{*}{\Rightarrow}_G w_i \ldots w_{i+\delta(k)}$

The proof itself is done by induction on $\delta(k)$, $k \in GREEN_G(w)$. We investigate $k \in GREEN_G(w)$ arbitrary, but fixed where $i := Col(\mathcal{S}(k))$.

INDUCTION BASIS:

$\underline{\delta(k) = 0:}$

$(\mathcal{I}_1)$ holds since it holds for all green edges k with $\mathcal{S}(k) = \mathcal{D}(k)$ and generated by the preprocessing.

$(\mathcal{I}_2)$ holds since there exist no green edges k with $\delta(k) = 0$.

$\underline{\delta(k) = 1:}$

$(\mathcal{I}_1)$ k is single with $\delta(k) = 1$.

CASE 1: k is generated by the preprocessing.

$$\Longrightarrow (i)\ \left((\gamma_1(\mathcal{S}(k)))^{-1}, (\gamma_1(\mathcal{D}(k)))^{-1}\right) \in \mathcal{K}[2]$$

$$(ii)\ \overline{\gamma_2(\mathcal{S}(k))} = \gamma_2(\mathcal{D}(k))$$

$$\Longrightarrow (Yield(k))^{-1} \overset{\text{Def}}{=} (\gamma_1(\mathcal{S}(k)))^{-1} \cdot (\gamma_1(\mathcal{D}(k)))^{-1} \Rightarrow_G w_i w_{i+1}$$

CASE 2: k is generated by (2).

$$\Longrightarrow (\gamma_1(\mathcal{S}(k)))^{-1} \in \mathcal{K}[1] \text{ and } \exists k' \in GREEN_G(w) \text{ where}$$

$$(i)\ Col(\mathcal{S}(k')) = i + 1$$

$$(ii)\ \delta(k') = \delta(k) - 1 = 0 \Longrightarrow k' \text{ single} \overset{(\mathcal{I}_1)}{\Longrightarrow} (Yield(k'))^{-1} \Rightarrow_G w_{i+1}$$

$$(iii)\ Prod(\mathcal{S}(k)) \cdot Yield(k') \equiv_{\rho_G} \varepsilon \overset{(E_2)}{\Longrightarrow} (Yield(k'))^{-1} = (Prod(\mathcal{S}(k)))^R$$

$$\Longrightarrow (Yield(k))^{-1} \overset{\text{Def}}{=} (\gamma_1(\mathcal{S}(k)))^{-1} \Rightarrow_G w_i \cdot (Prod(\mathcal{S}(k)))^R$$
$$= w_i \cdot (Yield(k'))^{-1} \Rightarrow_G w_i w_{i+1}$$

We are complete since (1) generates no single edge at all and (3) generates no single edge fulfilling $\delta(k) = 1$.

$(\mathcal{I}_2)$ k composed with $\delta(k) = 1$

$\Longrightarrow$ each $m_j \in marks(k)$ is generated by (1)

$\Longrightarrow$ $marks(k) = linear(k)$

$\Longrightarrow$ $\forall m_j := (k'_j, k''_j) \in marks(k)$:

(i) $k'_j, k''_j \in GREEN_G(w)$

(ii) $\delta(k'_j) + \delta(k''_j) = \delta(k) - 1 \Longrightarrow \delta(k'_j) = \delta(k''_j) = 0 \Longrightarrow k'_j, k''_j$ single

(iii) (a) $Col(\mathcal{S}(k'_j)) = i \overset{(\mathcal{I}_1)}{\Longrightarrow} \left(Yield(k'_j)\right)^{-1} \Rightarrow_G w_i$

(b) $Col(\mathcal{S}(k''_j)) = i+1 \overset{(\mathcal{I}_1)}{\Longrightarrow} \left(Yield(k''_j)\right)^{-1} \Rightarrow_G w_{i+1}$

$\Longrightarrow \forall 1 \le j \le |marks(k)|$: $(Yield_{lin}(k,j))^{-1} \overset{\text{Def}}{=} \left(Yield(k'_j)\right)^{-1} \cdot \left(Yield(k''_j)\right)^{-1} \Rightarrow^2_G w_i w_{i+1}$

INDUCTION STEP: $\delta(k) > 1$

($\mathcal{I}_1$) Let k single.

CASE 1: (1) generates no single edge.

CASE 2: k is generated by (2)

$\Longrightarrow \exists k' \in GREEN_G(w)$ where

(i) $Col(\mathcal{S}(k')) = i+1$ and $\delta(k') = \delta(k) - 1$

$\overset{\text{Ind.}}{\Longrightarrow} \forall u \in Yield(k')$: $u^{-1} \overset{*}{\Rightarrow}_G w_{i+1} \dots w_{i+1+\delta(k)-1} = w_{i+1} \dots w_{i+\delta(k)}$

(ii) $\exists u \in Yield(k')$: $Prod(\mathcal{S}(k)) \cdot u \equiv_{\rho G} \varepsilon$

$\overset{(E_2)}{\Longrightarrow} u^{-1} = (Prod(\mathcal{S}(k)))^R$

$\Longrightarrow (Yield(k))^{-1} = (\gamma_1(\mathcal{S}(k)))^{-1}$

$\Rightarrow_G w_i (Prod(\mathcal{S}(k)))^R = w_i u^{-1} \overset{*}{\Rightarrow}_G w_i w_{i+1} \dots w_{i+\delta(k)} = w_i \dots w_{i+\delta(k)}$

CASE 3: k is generated by (3).

$\Longrightarrow$ (i) $\left((\gamma_1(\mathcal{S}(k)))^{-1}, (\gamma_1(\mathcal{D}(k)))^{-1}\right) \in \mathcal{K}[2]$, $\overline{\gamma_2(\mathcal{S}(k))} = \gamma_2(\mathcal{D}(k))$

(ii) $\exists k' \in GREEN_G(w)$ where $Col(\mathcal{S}(k')) = i+1$,
$Col(\mathcal{D}(k')) = i + \delta(k) - 1$, and $\delta(k') = \delta(k) - 2$

$\overset{\text{Ind.}}{\Longrightarrow} \forall u \in Yield(k')$: $u^{-1} \overset{*}{\Rightarrow}_G w_{i+1} \dots w_{i+1+\delta(k)-2} = w_{i+1} \dots w_{i+\delta(k)-1}$

(iii) $\exists v := v_1 v_2 v_3 \in Yield(k')$:

(a) $Prod(\mathcal{S}(k)) \cdot v_1 \equiv_{\rho G} \varepsilon \overset{(E_2)}{\Longrightarrow} v_1^{-1} = (Prod(\mathcal{S}(k)))^R$

(b) $(v_3 \cdot Prod(\mathcal{D}(k)))^R \equiv_{\rho G} \varepsilon \overset{(E_2)}{\Longrightarrow} v_3^{-1} = (Prod(\mathcal{D}(k)))^R$

(c) $v_2 = \varepsilon$, since k single

$\Longrightarrow (Yield(k))^{-1} \overset{\text{Def}}{=} (\gamma_1(\mathcal{S}(k)))^{-1} \cdot (\gamma_1(\mathcal{D}(k)))^{-1}$

$\Rightarrow_G w_i (Prod(\mathcal{S}(k)))^R \cdot (Prod(\mathcal{D}(k)))^R \cdot w_{i+\delta(k)}$

$= w_i v_1^{-1} v_3^{-1} w_{i+\delta(k)} = w_i v^{-1} w_{i+\delta(k)}$

$\overset{*}{\Rightarrow}_G w_i w_{i+1} \dots w_{i+\delta(k)-1} w_{i+\delta(k)} = w_i \dots w_{i+\delta(k)}$

($\mathcal{I}_2$) Let k composed.

Here, each mark of k has to be investigated separately. Therefore, let $m_j \in marks(k)$ arbitrary, but fixed the mark under consideration.

CASE 1: m_j is generated by (1).

$\Longrightarrow m_j := (k', k'') \in linear(k)$

$\Longrightarrow Col(\mathcal{S}(k')) = i,\ Col(\mathcal{D}(k')) = i + r,\ Col(\mathcal{S}(k'')) = i + r + 1,$
$Col(\mathcal{D}(k'')) = i + \delta(k),\ \mathcal{D}(\mathcal{D}(k')) = \mathcal{S}(\mathcal{S}(k''))$ for some $r \in N_0$

$\overset{\text{Ind.}}{\Longrightarrow} \forall u_1 \in Yield(k')\ \forall u_2 \in Yield(k''):\ u_1^{-1} \overset{*}{\Rightarrow}_G w_i \dots w_{i+r}$
$u_2^{-1} \overset{*}{\Rightarrow}_G w_{i+r+1} \dots w_{i+\delta(k)}$

$\Longrightarrow \forall v \in Yield_{lin}(k, j)\ \exists u_1 \in Yield(k'), u_2 \in Yield(k'')$:

$v^{-1} \overset{\text{Def}}{=} u_1^{-1} \cdot u_2^{-1} \overset{*}{\Rightarrow}_G w_i \dots w_{i+\delta(k)}$

CASE 2: (2) generates only single edges.

CASE 3: m_j is generated by (3).

$\Longrightarrow k' := m_j \in nested(k)$

$\Longrightarrow \exists k'' \in GREEN_G(w)$:

(i) $Col(\mathcal{S}(k'')) = i + 1,\ Col(\mathcal{D}(k'')) = i + \delta(k) - 1$
$\overset{\text{Ind.}}{\Longrightarrow} \forall u \in Yield(k'') : u^{-1} \overset{*}{\Rightarrow}_G w_{i+1} \dots w_{i+\delta(k)-1}$

(ii) $\forall v_2 \in Yield(k')\ \exists v_1, v_3$: (a) $v = v_1 \cdot v_2 \cdot v_3 \in Yield(k'')$

(b) $Prod(\mathcal{S}(k)) \cdot v_1 \equiv_{\rho_G} \varepsilon$
$\overset{(E_2)}{\Longrightarrow} v_1^{-1} = (Prod(\mathcal{S}(k)))^R$

(c) $(v_3 \cdot Prod(\mathcal{D}(k)))^R \equiv_{\rho_G} \varepsilon$
$\overset{(E_2)}{\Longrightarrow} v_3^{-1} = (Prod(\mathcal{D}(k)))^R$

(iii) $\overline{\gamma_2(\mathcal{S}(k))} = \gamma_2(\mathcal{D}(k))$

$\Longrightarrow \forall u \in Yield(k, j)\ \exists v_2 \in Yield(k')$:

$u^{-1} \overset{\text{Def}}{=} (\gamma_1(\mathcal{S}(k)))^{-1} \cdot v_2^{-1} \cdot (\gamma_1(\mathcal{D}(k)))^{-1}$
$\Rightarrow_G w_i (Prod(\mathcal{S}(k)))^R \cdot v_2^{-1} \cdot (Prod(\mathcal{D}(k)))^R w_{i+\delta(k)}$
$= w_i v_1^{-1} \cdot v_2^{-1} \cdot v_3^{-1} w_{i+\delta(k)} = w_i v^{-1} w_{i+\delta(k)}$
$\overset{*}{\Rightarrow}_G w_i w_{i+1} \dots w_{i+\delta(k)-1} w_{i+\delta(k)} = w_i \dots w_{i+\delta(k)}$

Proof (of Lemma 5.2.)
Induction on $\delta := j - i$
INDUCTION BASIS:
$\delta = 0 \Longrightarrow W \Rightarrow_G w_i$
$\Longrightarrow W \to w_i \in P$ and $W \in \mathcal{K}[1]$
$\Longrightarrow$ claim because of the preprocessing

$\delta = 1 \Longrightarrow W \overset{*}{\Rightarrow}_G w_i w_{i+1}$

CASE 1: $W \in \mathcal{K}[2]$
$\Longrightarrow W = W_1 W_2$ with $(W_1, W_2) \to (w_i, w_{i+1}) \in P$
$\Longrightarrow$ There exists such an edge because of the preprocessing.

CASE 2: $W \in \mathcal{K}[1]^+$
Then, we have either $W = W_1 W_2$ with $W_1 \to w_i, W_2 \to w_{i+1} \in P$
$\overset{\delta=0}{\Longrightarrow}$ For both productions, there exists a green edge.

$\overset{(1)}{\Longrightarrow}$ claim

or it holds $W = W_1 \in \mathcal{K}[1]$ and there exists a $W_2 \in \mathcal{K}[1]$ with $W_1 \to w_i W_2 \in P$ and $W_2 \to w_{i+1} \in P$

$\overset{\delta=0}{\Longrightarrow}$ $\exists k' \in GREEN_G(w) : Col(\mathcal{S}(k')) = i+1,\ Col(\mathcal{D}(k')) = i+1,$
$W_2^{-1} \in Yield(k')$

$\overset{(2)}{\Longrightarrow}$ claim

INDUCTION STEP: $\delta > 1$

CASE 1: $|W| = 1$

$\Longrightarrow$ $W \in \mathcal{K}[1]$ und $W \overset{*}{\Rightarrow}_G w_i \dots w_{i+\delta}$

$\Longrightarrow$ $\exists W \to w_i X_1 \dots X_k \in P$ with $X_1 \dots X_k \overset{*}{\Rightarrow}_G w_{i+1} \dots w_{i+\delta}$

$\overset{\text{Ind.}}{\Longrightarrow}$ $\exists k \in GREEN_G(w) : (X_1 \dots X_k)^{-1} \in Yield(k),\ Col(\mathcal{S}(k)) = i+1,$
$Col(\mathcal{D}(k)) = i+\delta$

$\overset{(2)}{\Longrightarrow}$ claim

CASE 2: $|W| > 1$

(a) $W = \underbrace{W_1 \dots W_s}_{\in ED(\mathcal{K},\emptyset)} \underbrace{W_{s+1} \dots W_l}_{\in ED(\mathcal{K},\emptyset)}$

$\Longrightarrow$ $\exists j < \delta : W_1 \dots W_s \overset{*}{\Rightarrow}_G w_i \dots w_{i+j},$
$W_{s+1} \dots W_l \overset{*}{\Rightarrow}_G w_{i+j+1} \dots w_{i+\delta}$

$\overset{\text{Ind.}}{\Longrightarrow}$ $\exists k_1 \in GREEN_G(w) : Col(\mathcal{S}(k_1)) = i,\ Col(\mathcal{D}(k_1)) = i+j,$
$\exists k_2 \in GREEN_G(w) : Col(\mathcal{S}(k_2)) = i+j+1,\ Col(\mathcal{D}(k_2)) = i+\delta$

$\overset{(1)}{\Longrightarrow}$ claim

(b) $W = W_1 \underbrace{W_2 \dots W_{s-1}}_{\in ED(\mathcal{K},\emptyset)} W_s$ with $(W_1, W_s) \in \mathcal{K}[2]$

$\Longrightarrow$ $\exists 0 \le j < l \le \delta :$ (1) $W_2 \dots W_{s-1} \overset{*}{\Rightarrow}_G w_{i+j+1} \dots w_{i+l-1}$
(2) $(W_1, W_s) \to (w_i U, V w_{i+\delta}) \in P$ and
$UV \overset{*}{\Rightarrow}_G w_{i+1} \dots w_{i+j} w_{i+l} \dots w_{i+\delta-1}$

$\Longrightarrow$ $UW_2 \dots W_{s-1} V \overset{*}{\Rightarrow}_G w_{i+1} \dots w_{i+\delta-1}$

$\overset{\text{Ind.}}{\Longrightarrow}$ $\exists k \in GREEN_G(w) : Col(\mathcal{S}(k)) = i+1,\quad Col(\mathcal{D}(k)) = i+\delta-1,$
$(UV)^{-1} \in Yield(k)$

$\overset{(3)}{\Longrightarrow}$ claim

5.4. The Complexity

Each edge in $GREEN_G(w)$ is created only once. Different paths leading to the same edge are modelled by *marks*. For each $k \in GREEN_G(w)$, $|linear|(k) \le O(|CF(P)|^2 \cdot n)$ and $|nested|(k) \le O(|CF(P)|^2 \cdot n^2)$ hold because of their construction by the algorithm.

The preprocessing costs $O(|CF(P)|^2 \cdot n)$ operations. No more situations have to be examined, since only n transitions from one column to the next one exist.

The iteration investigates red edges s, s', where $d := Col(s') - Col(s)$ grows from 1 to $(n+2)$. Thus, each step is iterated at most $O(|CF(P)|^2 \cdot n^2)$ times.

Now, let s, s' be arbitrary, but fixed.

Step (1) applies to at most $|CF(P)|^2 \cdot n$ situations, where each costs a constant amount of operations.

Step (2) applies to at most $|CF(P)|$ situations, where each costs $O(n^4)$ to search a $v \in Yield(k)$ fulfilling $(Prod(s) \cdot v \equiv_{\rho_G} \varepsilon)$, if $marks(k)$ is realized as a matrix. (For each sign in the *Yield*, we have to investigate n^2 situations where each can cost $O(n^2)$ time because of the maximal length of *marks*.) All the other operations can be performed within $O(1)$ time.

Step (3) applies to at most $|CF(P)|^2$ situations, but the most time-consuming operation is, as for Step (2), the search for a suitable element inside a yield.

Thus, we get:

Theorem 5.1. Let $G \in CCFG(2)$ in generalized GNF, $w \in T^+$, and $n := |w|$. Our procedure solves the wordproblem for w relative to G in time $O(n^6)$ (more exactly: $O(|P|^8 \cdot n^6)$).

6. Conclusions

In this paper, we developed a completely new approach to analyse inputs w relative to $G \in CCFG(2)$. It essentially reduces a derivability relative to $CF(G)$ as well as a correct coupling inside a derivation tree to reducibility relative to a congruence relation. Its idea follows from the representation theorem of Shamir and its constructive proof in [8]. The complexity bound of our procedure is $O(n^6)$, where n is the length of the input. This is the same as we already know for $CCFG(2)$ (cf. [9]). Thus, the main point of our result is that it presents a completely new approach to analyse Coupled-Context-Free Grammars. Because of its natural relation to the definition of $CCFG$ (on the one hand, $CCFG(2)$ is defined over parentheses of rank 2, on the other hand, the algorithm essentially realizes the conditions in the definition of $ED(\mathcal{K}, T)$ with parentheses of rank ≤ 2), it seems to offer the best approach for the future to get still better bounds. In [10], the analysis of the structure of the graph presented here led to a representation theorem for $CCFG(2)$, which generalizes the well-known one for context-free languages by Chomsky–Schützenberger in [3]. Here, with this approach, we are able to show that already very slight modifications of the problem lead to NP-completeness. Future research should also investigate how to generalize our algorithm for $CCFG(l)$, $l > 2$.

References

[1] Abbeillé, A.: *Parsing French with Tree Adjoining Grammars: Some Linguistic Accounts.* In Proc. of the 12th Conf. on Computational Linguistics, 1988, 7-12.

[2] Bresnan, J., Kaplan, R., Peters, P., and Zaenen, A.: *Cross-serial Dependencies in Dutch.* Linguistic Inquiry **13**, 1982, 613-635.

[3] Chomsky, N. and Schützenberger, N.: *The Algebraic Theory of Context–Free Languages.* In "Computer Programming and Formal Systems" Amsterdam, North Holland, 1963, 118-161.

[4] Dassow, J. and Păun, G.: *Regulated Rewriting in Formal Language Theory.* Springer, 1989.

[5] Greibach, S.A. and Hopcroft, J.E.: *Scattered Context Grammars.* J. Comput. Syst. Sci. **3**, 1969, 232-247.

[6] Guan, Y., Hotz, G., and Reichert, A.: *Tree Grammars with multilinear Interpretation.* Technical Report, Univ. of Saarbrücken, 1992.

[7] Guan, Y.: *Klammergrammatiken, Netzgrammatiken und Interpretationen von Netzen.* PhD thesis, Univ. of Saarbrücken, 1992.

[8] Hotz, G. and Kretschmer, T.: *The Power of the Greibach Normal Form.* EIK **25**, 1989, 507-512.

[9] Hotz, G. and Pitsch, G.: *Fast Uniform Analysis of Coupled-Context-Free Grammars.* In Proc. of the 21^{th} International Colloquium on Automata, Languages, and Programming, 1994, 412-423.

[10] Hotz, G. and Pitsch, G.: *A Representation Theorem for Coupled-Context-Free Grammars.* In J. Almeida, G. Gomes, P. Silva (Eds.): Semigroups, Automata, and Languages, 1996, 137-144.

[11] Joshi, A.K., Levy, L.S., and Takahashi, M.: *Tree Adjunct Grammars.* J. Comput. Syst. Sci. **10**, 1975, 136-163.

[12] Joshi, A.K.: *An Introduction to Tree Adjoining Grammars.* In A. Manaster-Ramer(ed.), Mathematics of Language, John Benjamins Publishing Company, 1987.

[13] Kroch, T. and Joshi, A.K. : *The Linguistic Relevance of Tree Adjoining Grammars.* Technical Report MS-CIS-85-16, University of Pennsylvania, Philadelphia, 1985.

[14] Kroch, T.: *Unbounded Dependencies and Subjacency in a Tree Adjoining Grammar.* In A. Manaster-Ramer(ed.), Mathematics of Language, John Benjamins Publishing Company, 1987, 143-172.

[15] Pitsch, G.: *Analyse von Klammergrammatiken.* PhD thesis, Univ. of Saarbrücken 1993. also Technical Report 10/1994, Univ. of Saarbrücken, 1994.

[16] Pitsch, G.: *LL(k)-Parsing of Coupled-Context-Free Grammars.* Computational Intelligence **10**(4), 1993, 563-578.

[17] Pitsch, G.: *LR(k)-Coupled-Context-Free Grammars.* Information Processing Letters **55**, 1995, 349-358.

[18] Rosenkrantz, D.J.: *Matrix Equations and Normal Forms for Context–Free Grammars.* J. ACM **14**, 1967, 501-507.

[19] Shamir, E.: *A Representation Theorem for Algebraic and Context-Free Power Series in Noncommuting Variables.* Information and Control **11**, 1967, 239-254.

[20] Shieber, S.: *Evidence against Context–Freeness of Natural Language.* Linguistics and Philosophy **8**, 1986, 333-343.

[21] Schabes, Y. and Joshi, A.K.: *An Early-Type Parsing Algorithm for Tree Adjoining Grammars.* In Proc. of the 26^{th} Meeting of the Assoc. for Computational Linguistics, 1988.

[22] Vijay-Shanker, K. and Joshi, A.K.: *Some Computational Properties of Tree Adjoining Grammars.* In Proc. of the 23^{th} Meeting of the Assoc. for Computational Linguistics, 1985, 82-93.

A NP-Completeness

Above, we reduced the word problem for $w \in T^+$ relative to $G = (\mathcal{K}, T, P, S) \in CCFG(2)$ to an efficient path search in $NF\text{–}Graph_G(w)$. Analogously to the context-free case, we try to generalize this procedure for more general graphs. This can be described as follows:

Let $G = (V, E)$ a directed acyclic graph, where $\mathcal{B}$ resp. $\mathcal{F}$ are the begin resp. final node. The edges are marked by the mappings

$$\gamma_1 : E \longrightarrow \Sigma_1 \cup {\Sigma_1}^{-1} \cup \{\varepsilon\} \quad \text{and} \quad \gamma_2 : E \longrightarrow \Sigma_2 \cup {\Sigma_2}^{-1} \cup \{\varepsilon\}.$$

$\Sigma_1 := \{x_1, \ldots, x_r\}$ resp. $\Sigma_2 := \{y_1, \ldots, y_r\}$ are arbitrary alphabets, ${\Sigma_1}^{-1}$ resp. ${\Sigma_2}^{-1}$ the corresponding ones of formal inverses. γ_1 corresponds to our former mapping gained via $\varphi(t)$, $t \in T$, γ_2 to the naming of the productions as used by the algorithm. We use the congruence relations

$$\rho := \{(\sigma\sigma^{-1}, \varepsilon) \mid \sigma \in \Sigma_1\} \quad \text{and} \quad \tau := \{(\xi\xi^{-1}, \varepsilon) \mid \xi \in \Sigma_2\}.$$

which represent the correct nesting of sequences of parentheses from $(\Sigma_1 \cup \Sigma_1^{-1})$ resp. $(\Sigma_2 \cup \Sigma_2^{-1})$.

Question: Does there exist a path u in $NF\text{-}Graph_G(w)$ fulfilling $\mathcal{S}(u) = \mathcal{B}$, $\mathcal{D}(u) = \mathcal{F}$, $\gamma_1(u) \equiv_\rho \varepsilon$, and $\gamma_2(u) \equiv_\tau \varepsilon$?

Our former problem is obviously a special case of this one. In contrast to the context-free case, it holds:

Theorem A1. Given a directed acyclic graph with two markings on its edges, the decision problem presented is *NP*-complete.

Proof:
Obviously, there exists a nondeterministic linear time solution. To show the NP-hardness, we reduce 3SAT to a graph fulfilling the above conditions. Let the formula

$$\mathcal{FO} := \bigwedge_{i=1}^{n} \left(x_{i,1}^{\delta_{i,1}} \vee x_{i,2}^{\delta_{i,2}} \vee x_{i,3}^{\delta_{i,3}} \right)$$

where $\delta_{i,j} \in \{+1, -1\}$, n the number of the clauses, k the number of the variables x_l. Define $Var(\mathcal{FO}) := \{x_l \mid 1 \leq l \leq k\}$.

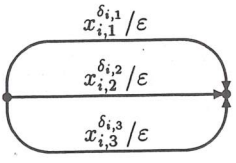
The constructed graph G consists of two subgraphs

G_A, where we guess a word which could satisfy $\mathcal{FO}$, and
G_B, where we analyse the guessed word whether it really satisfies $\mathcal{FO}$.

G results from these by overlying the final node of G_A with the begin node of G_B.

The construction of G_A

Each clause $C_i := (x_{i,1}^{\delta_{i,1}} \vee x_{i,2}^{\delta_{i,2}} \vee x_{i,3}^{\delta_{i,3}})$, $0 < i \leq n$, is transformed in the graph



All partial graphs obtained thereby are put in a sequence by identifying the end node for C_i with the begin node for C_{i+1}. Thus, we get G_A. Obviously, each path through G_A is marked in its second component by ε while the first component equals a guessed sequence of literals. We still have to test whether this sequence leads to a consistent assignment of the variables in $Var(\mathcal{FO})$. Then, we would have satisfiability of $\mathcal{FO}$.

Let $\Sigma_1 := Var(\mathcal{FO}) \cup \overline{Var(\mathcal{FO})}$, ${\Sigma_1}^{-1} := \{x_i^{-1} \mid x_i \in \Sigma_1\}$. Thus, the elements of Σ_1 are opening, the elements of ${\Sigma_1}^{-1}$ are closing parentheses. We define

$$\rho := \left\{(x_i x_i^{-1}, \varepsilon) \mid x_i \in Var(\mathcal{FO}) \cup \overline{Var(\mathcal{FO})}\right\}.$$

Σ_2, ${\Sigma_2}^{-1}$, and τ are defined identically using y_i instead of x_i.

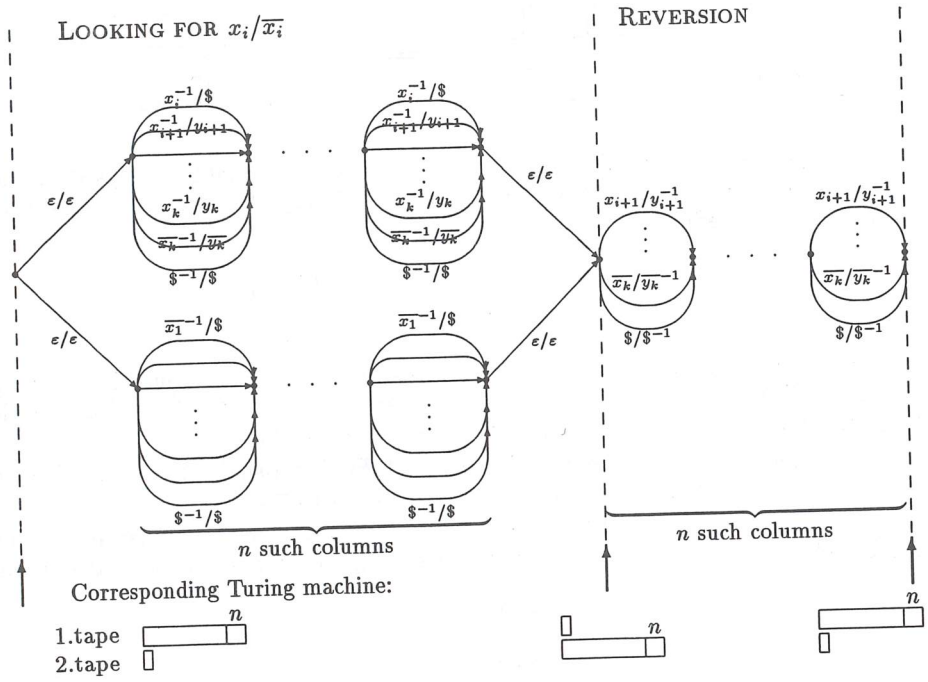


Figure 5 The ith subgraph of G_B

The construction of G_B

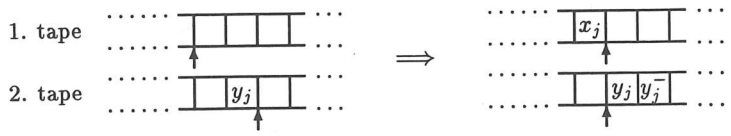
Following the guess of a satisfying assignment, we traverse $u\,|Var(\mathcal{FO})|$-often. During the ith traversal, we investigate the variable x_i, i.e. we test whether all occurrences of x_i in u are identical (either x_i or $\overline{x_i}$). Between every two occurrences of $x_i^{\delta_{i,j}}$ there are $x_l^{\delta_{l,r}}$ where $l \neq i$. They are stored since we need them for the investigation of the other variables. This storing is done by the second component of the marking.

The first and the second component of the marking can be imagined as two different tapes of a turing machine: when encountering x_i^{δ} at the end of the tape during the investigation of the variable x_i^{δ}, $(x_i^{\delta})^{-1}$ is written on this tape (because of ρ, this means shortening the content of the tape by one symbol).

When encountering a different variable, we do the same on the first tape (in order to see a new input symbol in the next step). Additionally, the variable found is written on the second tape.

Besides, we have to guarantee that our guess contains only x_i or $\overline{x_i}$, not both of them. Therefore, the analysis of the ith variable is done in two separate subgraphs (one for x_i and one for $\overline{x_i}$) which are identical up to the occurrences of x_i resp. $\overline{x_i}$ and which do not show any connecting edge.

Following this analysis, the resulting word modulo $\{(x_i x_i^{-1}, \varepsilon)\}$ resp. $\{(\overline{x_i}\overline{x_i}^{-1}, \varepsilon)\}$ is obviously written in inverse order on the second tape. Before going on to the variable $i+1$, this is moved back in correct order on the first tape (analogue procedure to above), i.e.



In order to investigate always a word of length n, the variables analysed are not only removed, but substituted by '\$'s. Only the last step of the analysis substitutes each \$ by ε. We define

$$\Sigma_1 := \Sigma_1 \cup \{\$\},\ {\Sigma_1}^{-1} := {\Sigma_1}^{-1} \cup \{\$^{-1}\},$$

$$\Sigma_2 := \Sigma_2 \cup \{\$\},\ {\Sigma_2}^{-1} := {\Sigma_2}^{-1} \cup \{\$^{-1}\}.$$

The subgraph of G_B analysing the ith variable is shown in Figure 5. Thus, the mappings $\gamma_1 : E \longrightarrow \Sigma_1 \cup {\Sigma_1}^{-1}$ and $\gamma_2 : E \longrightarrow \Sigma_2 \cup {\Sigma_2}^{-1}$ are uniquely defined. In addition, the begin node of G_A is denoted by $\mathcal{B}$, the final one of G_B by $\mathcal{F}$.

The graph $G := G_A \cdot G_B$ obviously fulfills that $\mathcal{FO}$ is satisfiable if and only if there is a path u in G from $\mathcal{B}$ to $\mathcal{F}$ where $\gamma_1(u) \equiv_\rho \varepsilon$ and $\gamma_2(u) \equiv_\tau \varepsilon$.

To be complete, the proof has to be independent of the number of variables. This can be achieved by coding the ($\leq 3n$) variables using an alphabet of constant size, e.g.

$$x_1 \mapsto a\$,\ x_2 \mapsto aa\$,\ \ldots$$

$$\overline{x_1} \mapsto \overline{a}\$,\ \overline{x_2} \mapsto \overline{aa}\$,\ \ldots$$