An Efficient Online Parser for Contextual Grammars with at Most Context–Free Selectors

Karin Harbusch

Univ. Koblenz–Landau, Computer Science Dept., harbusch@uni-koblenz.de

Abstract. Here we explore an efficient parser for a variant of contextual grammars (CGs) which is available via the internet. Contextual Grammar with context–free selectors allows for a simple representation of many non-context–free phenomena such as the copy language. Up to now, this specific type of selectors has hardly ever been addressed because even for simpler types no efficient parsing algorithms were known. In this paper, we describe a new polynomial parser which is based on the Earley algorithm. Furthermore we illustrate the linguistic relevance of Contextual Grammars with context–free selectors.

1 Introduction

Contextual Grammars (CGs) were originally introduced by Solomon Marcus [Marcus 1969] as “intrinsic grammars” without auxiliary symbols, based only on the fundamental linguistic operation of inserting words into given phrases according to certain contextual dependencies. The definition of CGs is simple and intuitive. Contextual Grammars include contexts, i.e., pairs of words, associated with selectors (sets of words). A context can be adjoined to any associated element in the selector (selector element). In this way, starting from a finite set of words (axioms), the language is generated.

For many variants of CG, a wide variety of properties has been studied. For instance, CGs do not fit into the Chomsky hierarchy (see, e.g., [Ehrenfeucht et al. 1997] for a recent overview of formal properties of CG). Concerning applications, it can be shown that this formalism provides an appropriate description of natural languages (cf. [Marcus et al. 1998]) such as the language $L_3 = \{a^n b^n c a^n b^m | n, m \geq 1 \}$, which is not a context–free language. $L_1$ circumscribes phenomena in a dialect spoken around Zurich (Switzerland) [Shieber 1985], which allows constructions of the form NP$^n_a$ NP$^m_g$ NP$^n_a$ 1 NP$^m_g$.$^1$

However, a surprising limitation of CGs with maximal global (Mg) use of selectors is discussed in [Marcus et al. 1998]. There exist center–embedded structures that cannot be generated by such a grammar even if regular selectors are imposed (cf. $L_3$ in Section 3). This limitation has led to the study of the variant of Contextual Grammars with context–free selectors which encompasses this shortcoming. Our claim is that CGs with context–free selectors provide a simple and

$^1$ Here NP$^n_a$ stands for an accusative noun phrase and NP$^m_g$ for a dative one.
adequate specification of a wide variety of linguistic phenomena as the selectors may range from linear to context–free and allow for a concise specification of linguistic pattern.

In order to apply all linguistically relevant classes of CGs for natural language processing an efficient parser has to be provided. In [Harbusch 1999], a polynomial parser for CGs with finite, regular and context–free selectors is outlined. Here we describe how to overcome its shortcoming of dealing with selector sets at run time. We propose to rewrite the selector language by an equivalent context–free grammar. All exponents in the infinite selector set, i.e., the Kleene star as well as up to two corresponding identical numerical ones, are couched into recursive rules. This behaviour reduces the grammar size and accordingly the average runtime. Furthermore, this paper focuses on the description of an online parser for this grammar type.

In the following section, Contextual Grammars are defined. In Section 3, the linguistic relevance of Contextual Grammars with context–free selectors is discussed. In Section 4, the online parser is presented. The paper ends by addressing some open questions and future work.

2 The Formalism of Contextual Grammars

In this section, we introduce the class of grammars we shall investigate in this paper. Here we adopt the terminology of [Marcus et al. 1998].

As usual, given an alphabet or vocabulary $V$, we denote by $V^{*}$ the set of all words or strings over $V$, including the empty string which is denoted by the symbol “$\lambda$”. The set of all non–empty words over $V$, hence $V^{*} - \{ \lambda \}$, is denoted by $V^{+}$. The length of $x \in V^{*}$ is depicted as $|x|$.

A Contextual Grammar (with choices) is a construct $G = (V, A, \{(S_1, C_1), \ldots, (S_n, C_n)\})$, $n \geq 1$, where $V$ is an alphabet, $A$ is a finite language over $V$, $S_1, \ldots, S_n$ are languages over $V$, and $C_1, \ldots, C_n$ are finite subsets $V^{*} \times V^{*}$. The elements of $A$ are called axioms or starting words, the sets $S_i$ are called selectors, and the elements $(u, v) \in C_i$ contexts. The pairs $(S_i, C_i)$ define productions or context–selector pairs.

For our purposes, the following terminology is added. For all $S_i \in S_i$ and all pairs $(c_{ik}, c_{ikr}) \in C_i$ ($1 \leq k \leq \text{cardinality of } C_i$) and $(S_i, C_i)$ is a production of the CG $G$, $(S_i, (c_{ik}, c_{ikr}))$ is called a context-selector pair–element; $c_{ik}$ is called a left context and $c_{ikr}$ a right context of the selector element $s_i$.

The direct derivation relation on $V^{*}$ is defined as $x \Rightarrow_{in} y$ iff $x = x_1 x_2 x_3, y = x_1 u x_2' x_3$, where $x_2 \in S_i, (u, v) \in C_i$, for some $i, 1 \leq i \leq n$.\footnote{The reader is also referred to [Martín–Vide et al. 1995], [Ilie 1996], [Ehrenfeucht et al. 1997], [Kudlek et al. 1997], [Marcus 1997], [Päun 1997], [Ehrenfeucht et al. 1998] for discussions of CG variants and their properties.}

\footnote{The index $i$ distinguishes the operation from $\Rightarrow_{ex}$, i.e. the external derivation (cf. the original definition by [Marcus 1969]) where the context is adjoined at the
If we denote the reflexive and transitive closure of the relation \( \rightarrow \) by \( \rightarrow^* \), the language generated by \( G \) is \( L_{in}(G) = \{ z \in V^* | w \rightarrow^*_m z, \text{ for some } w \in A \} \).

Two variants of the relation \( \rightarrow^*_m \) are defined as follows:

- \( x \rightarrow_{MI} y \) (maximal local mode) iff \( x = x_1 x_2 x_3, y = x_1 u x_2 v x_3, x_2 \in S_i, (u, v) \in C_i \), for some \( i, 1 \leq i \leq n \), and there are no \( x'_1, x'_2, x'_3 \in V^* \) such that \( x = x'_1 x'_2 x'_3, x'_2 \in S_i \), and \( |x'_1| \leq |x_1|, |x'_3| \leq |x_3|, |x'_2| > |x_2| \);

- \( x \rightarrow_{MG} y \) (maximal global mode) is defined the same as \( MI \) but here all selectors are regarded (\( x'_2 \in S_j, 1 \leq j \leq n \), the number of selectors).

For \( \alpha \in \{ MG, MI \} \) we denote: \( L_{\alpha}(G) = \{ z \in V^* | w \rightarrow^*_m z, \text{ for some } w \in A \} \).

In the next section, an example illustrating the selection according to the three derivation definitions is outlined.

If in a grammar \( G = (V, A, \{(S_1, C_1), \ldots, (S_n, C_n)\}) \) all selectors \( S_1, \ldots, S_n \) are languages in a given family \( F \), then we say \( G \) is a Contextual Grammar with \( F \) choice or with \( F \) selection. The families of languages \( L_{\alpha}(G) \), for \( G \) a Contextual Grammar with \( F \) choice, are denoted by \( CL_{\alpha}(F) \), where \( \alpha \in \{ in, MI, MG \} \).

### 3 Linguistic Relevance of CG with Context–Free Selectors

In [Marcus et al. 1998], the appropriateness of CGs for the description of natural languages is outlined. This property basically results from the fact that the grammar writer is able to straightforwardly describe all the usual restrictions appearing in natural languages. These statements require often more powerful formalisms than Context–Free Grammars (cf. reduplication, crossed dependencies, and multiple agreement).

For instance, the language \( L_2 = \{ xx | x \in \{ a, b \}^* \} \), which duplicates words of arbitrary length (copy language), allows for the construction of compound words of the form string–of–words–o–string–of–words as in Bambara, a language from the Mande family in Africa [Culy 1985]. Furthermore, the non-context–free language \( L_1 \) introduced in the introduction (crossed dependencies) circumscribes phenomena in a dialect spoken around Zurich (Switzerland) [Shieber 1985]. All these languages can be specified, e.g., with CGs with regular selectors under the assumption of \( MG \) (cf. [Marcus et al. 1998])\(^4\).

However, a surprising limitation of Contextual Grammars with maximal global use of selectors was briefly mentioned in the introduction. As stated in [Marcus et al. 1998] some so–called center–embedded structures such as \( L_3 = \)

ends of the derived words: \( x \Rightarrow_{cw} y \) if \( y = u x v \) for \( (u, v) \in C_i, x \in S_i, \) for some \( i, 1 \leq i \leq n \). We do not investigate \( \Rightarrow_{cw} \) here. [Päun & Nguyen 1980] proposed internal derivations.

\(^4\) \((\{ a, b, c \}, \{ c \}, \{ \{ a b \}^* \}, \{ \{ a, a \}, \{ b, b \} \}) \) yields the copy language; \((\{ a, b, c, d \}, \{ abcd \}, \{ ab^3 c \}, \{ (a, c) \}, \{ bc^2 d, \{ b, d \} \}) \) obtains crossed dependencies. Notice that for both grammars holds that the languages they produce are the same under the assumption of \( MI \) and \( MG \), respectively.
\{a^nb^mc^ma^n | n, m \geq 1\} cannot be generated. Note that \(L_3\) is a linear language in Chomsky’s sense and that it belongs to the families \(CL_{\alpha}(FIN)\) and \(CL_{\alpha}(FIN)\). However, \(L_3\) is not in the family \(CL_{\alpha}(REG)\).\footnote{Here \(CL_{\alpha}(F), \alpha \in \{in, Mg, MI\}, F \in \{FIN, REG\}\) denotes the contextual language where the selectors belong to the family of regular (REG) or finite (FIN) languages and the recursion definition is unrestricted (in) or selects maximal adjoinings (Mg/Mg, i.e. the longest element of the same selector/all selectors); CF refers to the family of context-free languages. For the formal definitions of FIN, REG, CF see, e.g., [Rozenberg & Salomaa 1997].}

In the following we explore the class of contextual languages with context-free selectors. Note here that CGs with context-free selectors yield in fact the full range of finite, regular and context-free selectors. Consequently, they cover all the languages mentioned in the beginning of this section. Furthermore, this adequacy constraint imposed on the process of grammar writing reduces the average runtime of the parser. For reasons of space we only investigate the question here whether Contextual Grammars with context-free selectors are able to capture \(L_3\). The grammar \(G_{cf} = \{(a, b, c), \{bcb\}, \{(a^ncb^mc^ma^n | n \geq 0), \{(a, a)\}, (b^mc^ma^n | n \geq 0), \{(b, b), (ac, ca)\}\) generates \(L_3\) under the assumption of \(Mg\). For an illustration how CG essentially works, all accepted input strings of \(G_{cf}\) up to the length 11 are outlined here:

The axiom \(bcb\) derives \(bcbcb\) or \(acbcbca\) according to \(in, Mg, MI\) because \(b^1c^3\) is the only applicable selector. For the two resulting strings the following holds:

- The string \(bcbcb\) derives \(bcbcbcb\) and \(acbcbcbca\) according to \(MI, Mg\) (here \(x_1 = \lambda, x_2 = b^1c^3b^1, x_3 = \lambda; x'_1 = b, x'_2 = b^2c^0b^1, x'_3 = b\) is suppressed as it would produce the same string; according to \(in\), furthermore \(babcacab\), \(baacbcbca\) (i.e., \(x'_1 = \lambda, x'_2 = b^3c^0b^0, x'_3 = b\) would be produced; for reasons of space, we omit their further consideration here).

- The string \(acbcbca\) derives \(aacacbcca\) according to \(Mg\) (here \(x_1 = \lambda, x_2 = a^1c^3b^1d^0a^0, x_3 = \lambda; x'_1 = a, x'_2 = a^0b^0c^0b^0, x'_3 = a\) is suppressed as it would produce the same string; according to \(in\), furthermore \(acbcbcbca, acbcbcbca, acbcbcbca\) would be produced (i.e., \(x'_1 = \lambda, x'_2 = b^1c^3b^1, x'_3 = a, x'_4 = acb, x'_5 = b^2c^0b^0, x'_6 = bca\); again, we omit their further consideration).

So, for the three strings \(bcbcbcb, acbcbcbca\) and \(acbcbcbca\) the following holds:

- The string \(bcbcbcb\) derives \(bcbcbcbcb\) and \(acbcbcbcbca\) according to \(Mg\) with the line of argumentation as before.

- The string \(acbcbcbca\) derives \(acbcbcbcbca\) according to \(Mg\) (here \(x_1 = \lambda, x_2 = a^1c^3b^2c^0\lambda, x_3 = \lambda, e.g., x'_1 = a, x'_2 = a^0c^2b^2c^0b^0, x'_3 = a\) or \(x'_1 = acb, x'_2 = b^2c^0b^0, x'_3 = bcb\) are suppressed as they would produce the same strings as \(acbacabcbca, acbacabcbca, acbacabcbca\) according to \(in\).

- The string \(acbcbcbca\) derives \(acbcbcbca\) according to \(Mg\) (\(x_1 = \lambda, x_2 = a^2b^0c^0b^1c^2a^2, x_3 = \lambda, e.g., x'_1 = a, x'_2 = a^1c^0b^0c^0b^1, x'_3 = a, x'_4 = aa, x'_5 = bcb, x'_6 = caa\).
Again, the string \( bbbbbbb \) derives \( bbbhebbbbb \) and \( acbbbebbbbb \) under the assumption of \( Mg \) according to the line of argumentation as before.

4 An Earley–based Parser for CGs

CG parsing is addressed, e.g., in [Ilie 1996] or [Boulier 2001]. In [Ilie 1996] External Contextual Grammars (i.e., CGs where the contexts can be added only at the end of the current string) are studied and it has been shown that this variant is basically parsable in polynomial time. In [Boulier 2001] a subclass of CGs is studied which can be translated into RCGs (Range Concatenation Grammars). For the resulting grammar it can be shown that parsing is polynomial. However some linguistically relevant variants cannot be covered by the transformation process. A more general approach is presented in [Harbusch 1999] or [Harbusch 2000]. Here, the basic idea of a polynomial parser for CGs with linear, regular, and context-free selectors is outlined. In the following, we extend this parser with respect to efficiency and describe the features of an online version through the internet.

First, a sketch of the intertwined two–level Earley–based\(^6\) parser is presented for Contextual Grammars with finite, regular or context–free selectors on the basis of the three derivation definitions \( in, Ml, Mg \).

In the following, the online parser becomes essentially extended in comparison to the algorithm described in [Harbusch 1999] or [Harbusch 2000]. Here the transformation of the selectors into a grammar is no more dependent on the length of the input string. From this fact the shortcoming arises that the grammar transformation has to be performed during the run time. Furthermore, the number of rules could grow dramatically because all accepted input strings up to the length of the currently considered input were enumerated. In the variant presented here a Context–Free Grammar is exploited to yield the acceptable selector pairs. We’ll show that this is more efficient than the comparison between strings. The online parser provides an automatic transformation of specified languages into a finite Context–Free Grammar licencing infinite selector languages. Furthermore this routine is a test whether the grammar is at most context–free. More powerful specifications are rejected. Before we address this transformation we describe the components of the CG parser.

4.1 The Components of the Parser

Our parser consists of two passes. Basically, in the first pass all individual contexts and selectors are identified and stored in items denoting the left and right boundary of these fragments. This task is performed by an ordinary Earley parser called \( FRAG \) (compute \( FRAG \) ments) on the basis of a Context–Free Grammar

\(^6\) The Earley algorithm is adopted here as it basically avoids normal–form transformations such as the elimination of \( \epsilon \) rules (rules such as \( X \rightarrow \epsilon \)) which are highly appreciated to state empty selectors and contexts, respectively.
with rules \((\text{sel}_i \rightarrow s_i)^7\), \((\text{con}_{ikl} \rightarrow c_{ikl}),(\text{con}_{ikr} \rightarrow c_{ikr})\) for all selector elements \(s_i \in S_i\) and all its context pairs \((c_{ikl}, c_{ikr}) \in C_i\) (\(\text{con}_{ikl}, \text{con}_{ikr}\) are nonterminals of the according Context-Free Grammar). All these items are used in the second-phase Earley-parser \((\text{PROCO} \rightarrow \text{PROduction COmination})\) in order to check the only context-free rule-type \((\text{sel}_i \rightarrow \text{con}_{ikl} \text{sel}_i \text{con}_{ikr})\), i.e., the identification of a derivation step of a CG. Notice that both components reuse the same implemented procedures “\text{PREDICT}”, “\text{SCAN}” and “\text{COMPLETE}” of the basic Earley algorithm. Any procedure is parametrized by the currently considered grammar and the input string (parameters may differ in the individual \textit{reruns} which result from the intertwined definition). All item lists are always available for prediction, scanning and completion\(^8\).

In order to become able to identify context-selector pairs after the elimination of contexts in the input string\(^9\) the two phases run intertwined (\textit{reruns}). Besides continuing the iteration in PROCO, for each successfully applied rule \((\text{sel}_i \rightarrow \text{con}_{ikl} \text{sel}_i \text{con}_{ikr})\), the input string where \(\text{con}_{ikl}\) and \(\text{con}_{ikr}\) are eliminated is handed back to the CG parser (i.e., FRAG and PROCO). It is important to note that the numbering in the newly build input strings remains the same so that the parsers in both phases can reuse all previously computed results. The strings to be erased are only marked to be empty by the new terminal \(\epsilon\). The rules \((q \rightarrow \epsilon q)\) and \((q \rightarrow q \epsilon)\) eliminate all occurrences of \(\epsilon\) and represents that the selector element covers the eliminated context as well.

In the worst case the space complexity of this acceptor is \(O(n^4)\) and the run time is \(O(n^5)\) for the ordinary recursion definition \(in\). The computation according to \(Mg, MI\) and parsing according to \(in, MI, Mg\), i.e., the computation of a condensed representation of all derivations, costs at most \(O(n^6)\) space and \(O(n^9)\) time units (see, e.g., [Harbusch 1999]).

The online parser provides the selection of predefined grammars beside the possibility of specifying the user’s own grammars. As output the transformed Context-Free Grammars, the initial item lists, the final item lists, a list of items causing reruns and the enumeration of all individual parses are presented in individual windows. With the decision of showing individual parses for reasons of readability by a human reader, the online parser could run exponentially in

\(^7\) Notice that this is exactly the point where the infinite length of selectors lead to the enumeration of accepted input strings or a grammatical representation of these strings. In Section 4.2, a more efficient alternative to circumscribe an infinite set \(S_i\) is presented.

\(^8\) As the parser is implemented in JAVA, according to the independence of processing, all these procedures run as individual Threads on possibly different machines. Actually, we use a 14x336 MHz 8-slot Sun Enterprise E4500/E5500 (sun4u) for our testing. However, the online parser runs on the local machine of the user. Accordingly, the performance may vary.

\(^9\) The following context-selector pairs provide an example of this necessity: \(((bd),\{(a, \lambda)\})), \(((d),\{(c, e)\})\). A possible derivation is \(bd \Rightarrow abd \Rightarrow abde\). Here the selector element \(bd\) is not a substring of the input string \(abde\). So the proper items which are constructed in the first phase are missing.
this final routine. If the core components become part of an natural language system this routine is not required. Accordingly the entire system is polynomial.

In Figure 1 the start and result window for the CG similar to Footnote 9 = \{(a, b, c, d, e), \{bd, a, \lambda\}, \{(a, \lambda), (a, b)\}, \{d\}, \{(c, e)\}\} is shown for the input string = abode. The representation of the item lists is omitted here for reasons of space (run the online parser with the provided grammar at this information).

4.2 Transformation of Infinite Selector Languages

In the following any given context-free selector w (i.e. the original rule (S \rightarrow w)) is wrapped up in a Context-Free Grammar which gives rise to a more efficient parser. For this endeavour the given selector language is inspected for substrings with exponents of the sort *, +, and n, respectively. These substrings are rewritten by recursive context-free rules. These procedures also allow to identify context-sensitive selectors (e.g., $a^n b^p c^n$) which are rejected as they cannot be parsed by the parser presented here.

First, the procedure KLEENETRANS with the parameters (string s, nonterminal nt), where initially s = w, nt = S, explores all substrings w' with exponent “*” or “+” or a numerical number that only occurs once in w. Its first occurrence is rewritten by a new nonterminal nt_i in the rules (nt \rightarrow w). The rule (nt_i \rightarrow w' without the exponent nt_i) is added. In case of exponent “*” (nt_i \rightarrow \lambda) is also deployed. KLEENETRANS is activated for the revised w and the substring w' which may contain nested exponents of the respective sort until no (more) exponents of these three sorts exist.

Second, the resulting rule set (which at least contains the original rule (S \rightarrow w)) is explored. Let us assume each rule has the form (lhs \rightarrow rhs). If in rhs two times the same exponent occurs, the recursive procedure CFTRANS with the parameters (string w, nonterminal nt) with w = rhs, nt = lhs runs in the following manner:

w is divided into $x_1 x_2^2 x_3 x_4^3 x_5$ (cf. pumping lemma; see, e.g., [Rozenberg & Salomaa 1997]) where $x_1$ does not contain any exponent. For the five new nonterminals nt2_1, ..., nt2_5, the following rules are constructed:

1. (nt \rightarrow x_1 nt_2_1 nt_2_2) + CFTRANS(x_5, nt_2_2),
2. (nt_2_1 \rightarrow nt_2_3 nt_2_1 nt_2_4) + CFTRANS(x_2, nt_2_3) + CFTRANS(x_4, nt_2_1),
3. (nt_2_1 \rightarrow nt_2_5) + CFTRANS(x_3, nt_2_5).

If no such exponents exist (nt \rightarrow w) is returned (end of the recursion).

Basically it is clear that both procedures terminate because always one or two exponents are rewritten. It can formally be shown that any context-free language can be rewritten. For reasons of space, we skip this proof here. As for the constructed rules in KLEENETRANS, it is directly obvious that the input word $w = a_1 x_1^{[i]} a_2 ... a_k x_k^{[i]} a_{k+1}$ (in $a_i$ no exponents of the form “*” or “+” occur (single numerical exponents are assumed here to be synonymously expressed by “*” or “+”, respectively; $1 \leq i \leq k + 1$); $x_j$ denotes a finite terminal string (1
The input string is: \( \text{a}^* \text{b}^* \text{c}^* \text{d}^* \text{e}^* \)

Notice, \(_*\) has technical reasons, it allows to identify terminals uniquely and should be ignored.

New input string for rerun initiated by item
(SEL_2-3, CON_L3 SEL_2-3 CON_R3, \(\lambda\), 2, 3, 4, \(\ldots\))
in item list 5 = \(\text{a} \text{b} \text{c}^* \text{d} \text{e}^*\)

According to the input string \(\text{a} \text{b} \text{c}^* \text{d} \text{e}^*\)
(SEL_1-1 -> CON_L1 SEL_1-1 CON_R1)

\[
\begin{align*}
\text{CON}_R1 & \rightarrow \lambda \\
(\text{SEL}_1-1 & \rightarrow \text{c} \text{d}^*) \\
(\text{d} & \rightarrow \text{d} \text{e}^*) \\
(\text{c} & \rightarrow \text{c}^* ) \\
\end{align*}
\]

and according to the rerun
(SEL_2-3 -> CON_L3 SEL_2-3 CON_R3)
(CON_R -> \(\text{e}^*\))

\[
\begin{align*}
(\text{SEL}_2-3 & \rightarrow \text{d}^*) \\
(\text{d} & \rightarrow \text{d}^*) \\
(\text{CON}_L3 & \rightarrow \text{c}^*) \\
(\text{c} & \rightarrow \text{c}^* ) \\
\end{align*}
\]

The input string is in the language.

---

Fig. 1. Start window and result window for a predefined CG
\[ \leq j \leq k \) is produced by a rule set consisting of \((S \rightarrow a_1 n t_1 \ldots a_k n t_k a_{k+1})\) and \((n t_i \rightarrow x_i n t_i)\) plus rules \((n t_i \rightarrow \lambda)\) in case of the exponent "**".

Concerning related exponents, with the pumping lemma and Chomsky–Schützenberger’s theorem (see, e.g., [Rozenberg & Salomaa 1997]), non-Dyck words are not context-free (e.g. \(\{\}\)). Accordingly, CFTRANS identifies recursively the outermost Dyck word \((x_1 x_3 x_5)\). Furthermore it checks whether Dyck words (i.e., nested exponential structures such as \(a^m[b^n e^n]^m\)) only lay inside a substring or behind the currently considered pair. By this method any construction with crossed brackets and more than two controlled brackets is rejected to be not context-free.

The runtime complexity of this procedure is linear with respect to the number of inspected rules. In general, it can only find finitely many exemplars of exponents in a rule of finite length to be rewritten. Any part without corresponding exponents is transformed into a single (no exponent in the source specification) or two (Kleene star repetition and initialization rewriting the Kleene star in the source specification) context-free rule(s) which costs \(O(1)\). The resulting grammar is much more condensed compared to the original parsing method, e.g., outlined in [Harbusch 1999] and hence impinging this representation on the online parser reduces the average runtime.

Figure 2 and 3 show the transformation of the grammar \(G_{cf}\). Notice that the online parser currently requires completely bracketed structures if exponents are specified. Furthermore, the variables in any exponent are interpreted as greater or equal to zero. Only for reasons of simplicity the user driven specification of two types of exponents is omitted. In the code both variants are yet tested.

5 Final Discussion

In this paper, we have addressed the linguistic relevance of Contextual Grammars with context-free selectors and described a more efficient version of the polynomial parser for CGs with \(CL_\alpha(F) \ (F \in \{FIN, REG, CF\}, \alpha \in \{in, MI, Mg\})\). The parser is implemented in JAVA and online available under the address:

http://www.uni-koblenz.de/~harbusch/CG-PARSER/welcome-cg.html

In the future, we’ll focus on the following two questions. Since we are especially interested in natural language parsing with CGs, we are going to build a Contextual Grammars with context-free selectors for English and German. Currently we are exploiting how to extract context-selector pairs from corpora. The heads are specified as features for selectors. The patterns, i.e., the contexts are extracted according to the significant examples in the corpus. On the theoretical side the properties of Contextual Grammars with context-free selectors will be studied in more detail.

References

**Self-defined Parsing Configuration**

**Warning: Applet Window**

Enter the input string in the following format:
- separate terminals by exactly one blank,
- no final blank,
- the empty string is indicated by \( \lambda \):

\[ a a c b b b c b b c a a \]

Enter the axioms in the following format:
- the axioms are separated by comma
- the terminals of an axiom are separated by blank,
- no final blank,

\[ b c b c b \]

Enter the context-selector pairs in the following format:
- put selectors and context pairs in brackets,
- separate selectors by blank
- separate left and right contexts by comma,

\[(a)^n c [b]^n c [b]^m c (a)^n), (a, a)\]
\[(b)^m c [b]^m, (b, b) (a c, c a)\]

Click here to send the input string and the grammar to the parser

**Fig. 2.** Example of a user-defined grammar
Fig. 3. Context–free transformations of the user–defined grammar in Figure 2


