Greibach Normal Form Transformation, Revisited

Robert Koch
Norbert Blum
Informatik IV, Universität Bonn
Römerstr. 164, D-53117 Bonn, Germany
email: blum@cs.uni-bonn.de

Abstract

We develop a direct method for placing a given context-free grammar into Greibach normal form with only polynomial increase of its size; i.e., we don't use any algebraic concept like formal power series. Starting with a cfg G in Chomsky normal form, we will use standard methods for the construction of an equivalent context-free grammar from a finite automaton and vice versa for transformation of G into an equivalent cfg G' in Greibach normal form. The size of G' will be $O(|G|^3)$, where $|G|$ is the size of G. Moreover, we show that it would be more efficient to apply the algorithm to a context-free grammar in canonical two form, obtaining a context-free grammar where, up to chain rules, the productions fulfill the Greibach normal form properties, and then to use the standard method for chain rule elimination for the transformation of this grammar into Greibach normal form. The size of the constructed grammar is $O(|G|^4)$ instead of $O(|G|^6)$, which we would obtain if we transform G into Chomsky normal form and then into Greibach normal form.

1 Introduction and definitions

We assume that the reader is familiar with the elementary theory of finite automata and context-free grammars as written in standard text books, e.g. [1, 4, 6, 9]. First, we will review the notations used in the subsequence.

A *context-free grammar* (cfg) G is a 4-tuple (V, Σ, P, S) where V is a finite, nonempty set of symbols called the *total vocabular*, $\Sigma \subset V$ a finite set of *terminal symbols*, $N = V \setminus \Sigma$ the set of *nonterminal symbols* (or *variables*), P a finite set of *rules* (or *productions*), and $S \in N$ is the *start sympol*. The productions are of the form $A \to \alpha$, where $A \in N$ and $\alpha \in V^*$. α is called *alternative* of A. $L(G)$ denotes the *context-free language* generated by G. The *size* $|G|$ of the cfg G is defined by

$$|G| = \sum_{A \to \alpha \in P} lg(A\alpha),$$

where $lg(A\alpha)$ is the length of the string $A\alpha$. Two context-free grammars G and G' are equivalent if both grammars generate the same language; i.e., $L(G) = L(G')$.

Let ε denote the empty word. A production $A \to \varepsilon$ is called *ε-rule*. A production $A \to B$ with $B \in N$ is called *chain rule*.

A *leftmost derivation* is a derivation where, at every step, the variable replaced has no variable to its left in the sentential form from which the replacement is made.

A cfg $G = (V, \Sigma, P, S)$ is in *canonical two form* if each production is of the form

i) $A \to BC$ with $B, C \in N \setminus \{S\}$,

ii) $A \to B$ with $B \in N \setminus \{S\}$,

iii) $A \to a$ with $a \in \Sigma$, or

iv) $S \to \varepsilon$.

A cfg G is in *Chomsky normal form* if G is in canonical two form and G contains no chain rule.

A cfg $G = (V, \Sigma, P, S)$ is in *extended Greibach normal form* if each production is of the form

i) $A \to aB_1B_2 \ldots B_t$ with $B_1, B_2, \ldots B_t \in N \setminus \{S\}, a \in \Sigma$,

ii) $A \to B$ with $B \in N \setminus S$,

iii) $A \to a$ with $a \in \Sigma$, or

iv) $S \to \varepsilon$.

A cfg G is in *extended 2-standard form* if G is in extended Greibach normal form and with respect to rules of kind (i), always $t \leq 2$.

A cfg $G = (V, \Sigma, P, S)$ is in *Greibach normal form* (Gnf) if G is in extended Greibach normal form and G contains no chain rule. G is in *2-standard form* (2-Gnf) if G is in extended 2-standard form and G contains no chain rule.

A *nondeterministic finite automaton* M is a 5-tuple $(Q, \Sigma, \delta, q_0, q_f)$, where Q is a finite set of states, Σ is a finite, nonempty set of input symbols, δ is a transition function mapping $Q \times \Sigma$ to 2^Q, $q_0 \in Q$ is the initial state, and $q_f \in Q$ is the finite state.

Given an arbitrary cfg $G = (V, \Sigma, P, S)$, it is well known that G can be transformed into an equivalent cfg G' which is in Gnf [3, 4, 6, 9]. But the usual algorithms possibly construct a cfg G', where the size of G' is exponential in the size of G (see [4], pp. 113–115 for an example). Given a cfg G without ε-rules and without chain rules, Rosenkrantz [8] has given an algorithm which produces an equivalent cfg G' in Gnf such that $|G'| = O(|G|^3)$. Rosenkrantz gave no analysis of the size of G'. For an analysis, see [4], pp. 129–130 or [7]. Rosenkrantz's algorithm uses formal power series.

We will develop a direct method for placing a given cfg into Gnf with only polynomial increase of its size; i.e., we don't use any algebraic concept like formal power series. Given any cfg G, in a first step the grammar will be transformed into an equivalent cfg G' in Cnf. Then, we will use standard methods for the construction of an equivalent cfg from an nfa and vice versa for transformation of G' into an equivalent cfg G'' in 2-Gnf. The size of G'' will be $O(|G'|^3)$. Moreover, we show that it would be more efficient to apply the algorithm to a cfg in canonical two form, obtaining a cfg in extended 2-standard form and then to use the standard method for chain rule elimination for the transformation of the grammar into 2-Gnf. The size of the constructed grammar is $O(|G|^4)$ instead of $O(|G|^6)$, which we would obtain if we transform G into Cnf and then into Gnf.

2 The method

Let $G = (V, \Sigma, P, S)$ be an arbitrary cfg in Cnf. Productions of type $A \to a$ with $a \in \Sigma$ already fulfill the Greibach normal form properties. Our goal is now to replace the productions of type $A \to BC$, $B, C \in N \setminus \{S\}$ by productions which fulfill the Greibach normal form properties. Let $B \in N \setminus \{S\}$. We have an interest in leftmost derivations of the form

$$B \Rightarrow a \text{ or } B \Rightarrow^*_{lm} B_0 B_1 \ldots B_t \Rightarrow a B_1 B_2 \ldots B_t,$$

where $a \in \Sigma$ and $B_0, B_1, \ldots, B_t \in N \setminus \{S\}$. Up to the last replacement, only alternatives from $(N \setminus \{S\})^2$ are chosen. The last replacement chooses for B_0 an alternative in Σ. Such a leftmost derivation is called *terminal leftmost derivation* and is denoted by

$$B \Rightarrow_{tlm} a \text{ and } B \Rightarrow^*_{tlm} a B_1 B_2 \ldots B_t, \text{ respectively.}$$

Let $L_B = \{a\alpha \in \Sigma(N \setminus \{S\})^* \mid B \Rightarrow^*_{tlm} a\alpha\}$. Our goal is to construct a cfg $G_B = (V_B, V, P_B, S_B)$ such that

a) $L(G_B) = L_B$, and

b) each alternative of a variable begins with a symbol in Σ.

N_B denotes the set of nonterminals of G_B; i.e. $N_B = V_B \setminus V$. Assume that for $B, C \in N \setminus \{S\}$, $B \neq C$, the grammars G_B and G_C are constructed such that

i) $N_B \cap N_C = \emptyset$, or

ii) each production with a variable from $N_B \cap N_C$ on the left side is contained in both set of rules P_B and P_C.

If we replace each production $A \to BC \in P$ by the set

$$\{A \to a\gamma C \mid a\gamma \text{ is an alternative of } S_B \text{ in } G_B\},$$

then we obtain a cfg $G' = (V', \Sigma, P', S)$ in Gnf, where

$$\begin{aligned}
V' = \; & V \cup \{N_B \mid B \in N \setminus \{S\}\}, \text{ and} \\
P' = \; & \{A \to a \in P \mid a \in \Sigma\} \\
& \cup \{A \to a\gamma C \mid A \to BC \in P \text{ and } a\gamma \text{ is an alternative of } S_B \text{ in } G_B\} \\
& \cup \textstyle\bigcup_{B \in N \setminus \{S\}} P_B \setminus \{S_B \to \alpha \mid \alpha \in \Sigma N_B^*\}
\end{aligned}$$

The conditions (i) and (ii) above ensure that in a derivation in G' no illegal changes between two grammars G_B and G_C are possible. It remains the construction of the cfg $G_B = (V_B, V, P_B, S_B)$, $B \in N \setminus \{S\}$. The central observation is that L_B is a regular set. Let

$$L_B^R = \{B_t B_{t-1} \dots B_1 a \mid a B_1 B_2 \dots B_t \in L_B\};$$

i.e., L_B^R is L_B reversed.

First, we will construct a nfa $M_B = (Q, V, \delta, B_B, \mathcal{S}_B)$ with $L(M_B) = L_B^R$. Using a standard method (see [4], pp. 55–56), it is easy to construct from M_B a nfa M'_B with $L(M'_B) = L_B$. For doing this, $\mathcal{S}_B$ will be the initial state, B_B will be the unique finite state, and we turn around the transitions of M_B. Then, we will use for each $B \in N \setminus \{S\}$ the nfa M'_B for the construction of a cfg $G'_B = (V_B, V, P'_B, \mathcal{S}_B)$ which fulfills

1. $L(G'_B) = L_B$,
2. G'_B is in Gnf. (Note that V is the terminal alphabet of G'_B.)
3. $\alpha \in \Sigma N_B^*$ for each production $\mathcal{S}_B \to \alpha \in P'_B$.
4. The right side of every other production begins with a variable of G; i.e., a symbol in $N \setminus \{S\}$.
5. $N_B \cap N_C = \emptyset$ for all $B, C \in N \setminus \{S\}$, $B \neq C$.

Now, G_B will be constructed from G'_B by the replacement of every rule of the form $D_B \to EC_B$ and $D_B \to E$, respectively in P'_B by a set of productions, analogously to the construction of P' from P. Altogether, we obtain the following method for the construction of the cfg $G_B = (V_B, V, P_B, \mathcal{S}_B)$.

(1) Define $M_B = (Q, V, \delta, B_B, \mathcal{S}_B)$ by

$$\begin{aligned} Q = & \quad \{A_B \mid A \in N\} \cup \{\mathcal{S}_B\}, \text{ and} \\ \delta(C_B, E) = & \quad \{D_B \mid C \to DE \in P\} \\ \delta(C_B, a) = & \quad \begin{cases} \{\mathcal{S}_B\} & \text{if } C \to a \in P \\ \emptyset & \text{otherwise} \end{cases} \end{aligned}$$

for all $C_B \in Q$, $E \in N \setminus \{S\}$, $a \in \Sigma$.

$L(M_B) = L_B^R$ can easily be proven by induction on the length of the terminal left derivation and on the length of the computation of the nfa, respectively.

(2) Define $M'_B = (Q, V, \delta', \mathcal{S}_B, B_B)$ by

$$\begin{aligned} \delta'(\mathcal{S}_B, a) = &\quad \{C_B \mid \{\mathcal{S}_B\} = \delta(C_B, a)\}, \text{ and} \\ \delta'(D_B, E) = &\quad \{C_B \mid D_B \in \delta(C_B, E)\} \end{aligned}$$

for all $D_B \in Q$, $E \in N \setminus \{S\}$ and $a \in \Sigma$.

Also $L(M'_B) = L_B$ can easily be proven by induction. Using the standard method for the construction of an equivalent cfg from a given nfa (see [4], pp. 61–62), we obtain the cfg G'_B.

(3) Define $G'_B = (V_B, V, P'_B, \mathcal{S}_B)$ by

$$\begin{array}{rll} V_B = & \{A_B & \mid A \in N \setminus \{S\}\} \cup \{\mathcal{S}_B\} \cup V, \text{ and} \\ P'_B = & \{\mathcal{S}_B \to aC_B & \mid C_B \in \delta'(\mathcal{S}_B, a) \text{ and} \\ & & (C_B \neq B_B \text{ or } \delta'(\{B_B\} \times N \setminus \{S\} \neq \emptyset)\} \\ & \cup\{\mathcal{S}_B \to a & \mid B_B \in \delta'(\mathcal{S}_B, a)\} \\ & \cup\{D_B \to EC_B & \mid C_B \in \delta'(D_B, E) \text{ and} \\ & & (C_B \neq B_B \text{ or } \delta'(\{B_B\} \times N \setminus \{S\} \neq \emptyset)\} \\ & \cup\{D_B \to E & \mid B_B \in \delta'(D_B, E)\} \end{array}$$

for all $D_B \in V'_B$, $E \in N \setminus \{S\}$, $a \in \Sigma$.

By inspection, Properties 1 – 5 can easily be proven. It is easy to show that $L(G'_B) = L_B$. For obtaining only productions fulfilling the Greibach normal properties with respect to the terminal alphabet Σ, simultaneously to all G'_B, $B \in N \setminus \{S\}$, we apply the trick used for the construction of P' from P again. Note that for all grammars G'_E, $E \in N \setminus \{S\}$, the alternatives of the start symbol $\mathcal{S}_E$ fulfill the Greibach normal form properties with respect to the terminal alphabet Σ.

(4) For each $B \in N \setminus \{S\}$ we define $G_B = (V_B, \Sigma, P_B, \mathcal{S}_B)$ by

$$\begin{array}{rl} P_B = & \{\mathcal{S}_B \to aC_B \mid \mathcal{S}_B \to aC_B \in P'_B\} \\ & \{\mathcal{S}_B \to a \mid \mathcal{S}_B \to a \in P'_B\} \\ & \cup \bigcup_{D_B \to EC_B \in P'_B} \{D_B \to \alpha C_B \mid \alpha \text{ is an alternative of } \mathcal{S}_E \text{ in } G'_E\} \\ & \cup \bigcup_{D_B \to E \in P'_B} \{D_B \to \alpha \mid \alpha \text{ is an alternative of } \mathcal{S}_E \text{ in } G'_E\} \end{array}$$

Note that $L(G_B) \neq L_B$, since the same trick was applied twice and hence, the terminal alphabet of G_B is Σ and not V. We have to add to P_B all productions in G_E with left side $\neq \mathcal{S}_E$. Since we use for each $E \in N \setminus \{S\}$ the grammar G_E for the construction of the cfg $G' = (V', \Sigma, P', S)$ and all these productions are added to P', we do not have to add these productions explicitely to P_B. For the same reasons, we have not to extend N_B explicitely.

By construction, the cfg G_B fulfills the Greibach normal form properties with respect to the terminal alphabet Σ for all $B \in N \setminus \{S\}$. Furthermore, for all $B, C \in N \setminus \{S\}$, $B \neq C$ the grammar G_B and G_C are constructed, such that

i) $N_B \cap N_C = \emptyset$, or

ii) each production with a variable from $N_B \cap N_C$ on the left side is contained implicitely in both set of rules P_B and P_C.

Since $L(G'_B) = L_B$ for all $B \in N \setminus \{S\}$, $L(G') = L(G)$ follows immediately. Furthermore, G' is in Gnf. Moreover, since every alternative α of the start symbol $\mathcal{S}_B$ of the grammars G'_B and G_B are of the form $\alpha = aE$, $a \in \Sigma, E \in N \setminus \{S\}$, the grammar G' is in 2-Gnf.

Let us analyze the size of G' in dependence of the size of G. Since each transition of M_B and hence, each transition of M'_B corresponds to a rule of P and vice versa, it is easy to see that $|G'_B| \leq 5|G|$ for all $B \in N \setminus \{S\}$. Constructing G_B from G'_B, every production $D_B \to EC_B$ and $D_B \to E$, respectively of P'_B is replaced by at most $|G'_E|$ rules. Hence, $|G_B| \leq |G'_B||G| = O(|G|^2)$. Similary, each production of type $A \to BC$ of P is replaced by at most $|G|$ productions. Note that for all $B \in N \setminus \{S\}$, the number of alternatives of the start symbol $\mathcal{S}_B$ of G_B is bounded by $|G|$. Altogether, we have obtained

$$|G'| \leq 5|G|^2 + \sum_{B \in N \setminus \{S\}} |G_B| = O(|G|^3).$$

Hence, we have proven the following theorem.

Theorem 1 *Let $G = (V, \Sigma, P, S)$ be a cfg in Cnf. Then there is a cfg $G' = (V', \Sigma, P', S)$ in 2-Gnf such that $L(G') = L(G)$ and $|G'| = O(|G|^3)$.*

Given an arbitrary cfg $G = (V, \Sigma, P, S)$, the usual algorithm for the transformation of G into Cnf can square the size of the grammar (see [4], pp. 102 for an example). No better algorithm is known. This observation leads directly to the following corollary.

Corollary 1 *Let $G = (V, \Sigma, P, S)$ be a cfg. Then there exists a cfg $G' = (V', \Sigma, P', S)$ in 2-Gnf such that $L(G') = L(G)$ and $|G'| = O(|G|^6)$.*

In [2], it is shown that for all $\epsilon > 0$ and sufficiently large n there is a context-free language CL_n with the following properties:

a) CL_n has a cfg of size $O(n)$.

b) Each chain rule free cfg for CL_n has size $O(\epsilon^3 n^{3/2-\epsilon})$.

3 Improving the size of the Gnf grammar

Harrison and Yehudai [5] have developed an algorithm which eliminates for a given cfg $G = (V, \Sigma, P, S)$ the ε-rules in linear time. Moreover, the constructed cfg $G' = (V', \Sigma, P', S)$ is in canonical two form and $|G'| \leq 12|G|$. For obtaining an equivalent cfg G'' in Cnf from G', it would suffice to eliminate the chain rules from G'. Hence, the expensive part of the transformation of an arbitrary cfg $G = (V, \Sigma, P, S)$ into Cnf is the elimination of the chain rules. The standard method for chain rule elimination computes for all $A \in N$ the set

$$W(A) = \{B \in N \mid A \Rightarrow^* B\}$$

and deletes the chain rules from P and adds the set

$$P(A) = \{A \to \alpha \mid \alpha \in V^+ \setminus N \text{ and } \exists B \in W(A) : B \to \alpha \in P\}$$

of productions to P. Note that $|P(A)| \leq |G|$. Hence, $|G''| = O(n^2)$.

Given an arbitrary cfg $G = (V, \Sigma, P, S)$, our goal is now to construct an equivalent cfg G' in Gnf such that $|G'| = O(|G|^4)$. Since the transformation of G into canonical two form enlarges the size of G at most by the factor 12, we can assume that G is already in canonical two form. First, we will show that an appropriate extension of our algorithm will produce an equivalent cfg $\bar{G}$ in extended 2-standard form. Then we will show that applying the standard method for chain rule elimination produces a cfg G' in 2-standard form with $L(G') = L(G)$ and $|G'| = O(|G|^4)$.

Let $G = (V, \Sigma, P, S)$ be a cfg in canonical two form. Then the chain rules already fulfill the properties for extended 2-standard form. Hence, the grammar $\bar{G} = (V', \Sigma, \bar{P}, S)$, where

$$\begin{array}{ll} V' = & V \cup \{N_B \mid B \in N \setminus \{S\}\}, \text{ and} \\ \bar{P} = & \{A \to a \in P \mid a \in \Sigma\} \\ & \{A \to B \in P \mid B \in N\} \\ & \cup\{A \to a\gamma C \mid A \to BC \in P \text{ and } a\gamma \text{ is an alternative of } \mathcal{S}_B \text{ in } G_B\} \\ & \cup \bigcup_{B \in N\setminus\{S\}} P_B \setminus \{\mathcal{S}_B \to \alpha \mid \alpha \in \Sigma N_B^*\} \end{array}$$

would be in extended 2-standard form, assumed that the grammars G_B, $B \in N \setminus \{S\}$ are constructed correctly. It remains to discuss, how to treat the chain rules during the construction of G_B. For doing this, we will extend the four steps of the algorithm in an appropriate manner.

(1) We add the following transitions to δ.

$$\delta(C_B, \varepsilon) = \{D_B \mid C \to D \in P\}$$

for all $C_B \in Q$.

(2) For the correct definition of M'_B, we have to add the following transitions to δ'.

$$\delta'(D_B, \varepsilon) = \{C_B \mid D_B \in \delta(C_B, \varepsilon)\}$$

for all $D_B \in Q$.

(3) The standard algorithm for the construction of a cfg from a given nfa add the following extra rules to P'_B.

$$\{D_B \to C_B \mid C_B \in \delta'(D_B, \varepsilon) \text{ and } (C_B \neq B_B \text{ or } \delta'(\{B_B\} \times N \setminus \{S\}) \neq \emptyset)\}$$

for all $D_B \in V'_B$.

Step 4 has not to be extended.

For the elimination of the chain rules $A \to B$ from $\bar{P}$, we add the set

$$\{A \to a\gamma \mid \exists B \in W(A) \text{ and } a\gamma \text{ is an alternative of } \mathcal{S}_B \text{ in } G_B\}$$

of productions to P'. Since $|W(A)| \leq |N|$ for all $A \in N$ and since every start symbol $\mathcal{S}_B$ of a grammar G_B has at most $|G|$ alternatives, this enlarges the size of $\bar{G}$ at most by an amount of $O(|G|^2)$. Now, we apply the standard method for chain rule elimination to G_B for all $B \in N \setminus \{S\}$. Note that for all $C_B \in N_B$, $|W(C_B)| \leq |N_B|$ and hence, $|W(C_B)| \leq |G|$. Furthermore, the number of distinct alternatives of variables in V_B is bounded by $O(|G|^2)$. Hence, the standard method for chain rule elimination enlarges the size of G_B at most by a factor $|G|$. Hence, $|G_B| = O(|G|^3)$. Altogether, we have obtained the following theorem.

Theorem 2 *Let $G = (V, \Sigma, P, S)$ be a cfg. Then there exits a cfg $G' = (V', \Sigma, P', S)$ in 2-Gnf such that $L(G') = L(G)$ and $|G'| = O(|G|^4)$.*

Remark: Given an ε-rule free cfg $G = (V, \Sigma, P, S)$, the transformation of G into canonical two form can enlarge the number of production considerably. If one does not wish to enlarge the number of productions in such a way, one can transform G directly into extended Greibach normal form. Let $A \to B\alpha$ be any production of G. We consider α as one symbol and built in Steps 1 and 2 nfa's $M_B = (Q, V_{ext}, \delta, B_B, \mathcal{S}_B)$ and $M'_B = (Q, V_{ext}, \delta', \mathcal{S}_B, B_B)$, respectively, where $V_{ext} = V \cup \{\alpha \mid \exists E \in V$ with $E\alpha$ is an alternative of an variable$\}$. Then we construct the cfg G'_B from M'_B. During Step 4, α will be considered as an element of V^*.

Acknowledgment: We thank Claus Rick for helpful comments.

References

[1] A. V. Aho, and J. D. Ullman, *The Theory of Parsing, Translation, and Compiling*, Vol. I: Parsing, Prentice-Hall (1972).

[2] N. Blum, More on the power of chain rules in context-free grammars, *TCS* **27** (1983), 287–295.

[3] S. A. Greibach, A new normal-form theorem for context-free, phrase-structure grammars, *JACM* **12** (1965), 42–52.

[4] M. A. Harrison, *Introduction to Formal Language Theory*, Addison-Wesley (1978).

[5] M. A. Harrison, and A. Yehudai, Eliminating null rules in linear time, *The Computer Journal* **24** (1981), 156–161.

[6] J. E. Hopcroft, and J. D. Ullman, *Introduction to Autmata Theory, Languages, and Computation*, Addison-Wesley (1979).

[7] A. Kelemenová, Complexity of normal form grammars, *TCS* **28** (1984), 299–314.

[8] D. J. Rosenkrantz, Matrix equations and normal forms for context-free grammers, *JACM* **14** (1967), 501–507.

[9] D. Wood, *Theory of Computation*, Harper & Row (1987).