The Hardest Context-free Language Revised

Julia Stoll

Report A/2002/4

ISBN 951-781-264-7

UNIVERSITY OF KUOPIO
Department of Computer Science and Applied Mathematics

P.O.Box 1627, FIN-70211 Kuopio, FINLAND

(intentionally left blank)

The Hardest Context-free Language Revised

Julia Stoll
julias@acm.org

Abstract

The Hardest Context-free Language L_0 is one context-free language (cfl) from which any cfl may be obtained by an inverse homomorphism [Gre73]. One key of proving that the Hardest Context-free Language L_0 exists is that every cfl L can be generated by a context-free grammar (cfg) G in Greibach Normal form (GNF) so that $L = L(G)$ [Gre65].

We present a new proof of the well-known result given in [Gre73] using the new method of transforming a cfg into 2-Greibach Normal form (2-GNF) in $O(|G|^3)$ presented in [BK99]. Revising the proof of the existence of L_0 enables us to predict the length of pairs of sentential forms. The result is that a pair of sentential forms s_k and s_{k+1}, $k \geq 1$, $S := s_1$, organized in blocks of "$x_i c y_i c z_i d$", $1 \leq i \leq n$, for generating L_0 have an upper bound and a lower bound in their length (if $\lambda \notin L$); where $|s_k|$ and $|s_{k+1}|$ are the length of a pair of sentential forms, then it holds $|s_k| \leq |s_{k+1}| \leq |s_k| + 2$.

Classification: F.4.3

1 Introduction

We present a new proof of the well-known result given in [Gre73]. One key of proving that the Hardest Context-free Language exists is that every cfl L can be generated by a cfg G into GNF so that $L = L(G)$ [Gre65]. A new method for transforming a cfg into 2-GNF is presented in [KB97, BK99]. The transformation method produces a 2-GNF in $O(|G|^3)$. Whereas the length of the righthand sides of a rule in set of rules P were unknown in the 'classical GNF' [Gre65], we know that these are restricted by transforming a cfg into 2-GNF. Of course, this fact has an impact on the construction of the Hardest Context-free Language. A pair of sentential forms s_k and s_{k+1} organized in blocks "$x_icy_icz_id$", $k \geq 1$, $S := s_1$, for generating L_0 have an upper bound and a lower bound in their length. If $|s_k|$ and $|s_{k+1}|$ are the length of a pair of sentential forms, then it holds $|s_k| \leq |s_{k+1}| \leq |s_k| + 2$.

In Sec. 2 we give the basic notations, definitions and well-known results. Different kind of GNF's are defined in Sec. 2.1. Semi-Dyck languages and their properties are introduced in Sec. 2.2. The definition of the Hardest Context-free Languages is given in Sec. 2.3. The new proof based on a 2-GNF is given in Sec. 3. Observations and properties derivated by the new proof are listed in Sec. 4. In Subsec. 4.2 our main result, Theorem 4.1, follows the Corollaries 4.1 and 4.2, as presented in Sec. 4.

2 Preliminaries

Let $G = (N, T, P, S)$ be a Chomsky-grammar, where N is a finite, non-empty alphabet of nonterminals, T is a finite, non-empty alphabet of terminals, P is a set of rules (or productions), and $S \in N$ is a start symbol. The G is context-free, iff all rules $r \in P$ are of the form $r : p \to q$, $p \in N$ $(|p| = 1)$, and $q \in (N \cup T)^*$. Let $G = (N, T, P, S)$ be a cfg and let $u', v' \in (N \cup T)^*$. It is said that u' directly derives v', written $u' \Rightarrow v'$, if there exists $u_1, u_2, v \in (N \cup T)^*$ and $A \in N$, such that $u' = u_1Au_2$, $v' = u_1vu_2$ and $A \to v \in P$. We write $\Rightarrow^*$ for the reflexive and transitive closure of $\Rightarrow$. The language generated by a grammar G is defined by $L(G) = \{w \in T^* | \ S \Rightarrow^* w\}$. A language L, generated by a cfg with $L = L(G)$, is called a context-free language (cfl) [Cho59, Sal73, Har78]. If L is a clf, then $init(L) = \{u | uv \in L$ for some $T^*\}$. If $w \in L(G)$, then the sequence $S \Rightarrow s_2 \Rightarrow \cdots \Rightarrow s_i \Rightarrow \cdots \Rightarrow s_n = w$, $1 \leq i \leq n$, is a derivation of the word w and the s_i's are called sentential forms, where S is also a sentential form indexed by 1. We define a norm $\|G\|$ on rules of a cfg $G = (N, T, P, S)$. Let $\|G\|$ be the longest righthand side of all rules r in P so that the norm is defined as $\|G\| := \{max(q) | p \to q, p \in N(|p| = 1)$, and $q \in (N \cup T)^*$ for all $r \in P\}$.

2.1 Greibach Normal forms

A cfg $G = (N, T, P, S)$ is in Greibach Normal form (GNF), iff it contains only rules of the form $A \to a\alpha$, where $a \in T$ and $\alpha \in (N \cup T)^+ \backslash \{S\}$. If $\lambda \in L$, then it is $S \to \lambda$ in P. Every cfl L may be generated by a cfg G in GNF so that $L = L(G)$ [Gre65, Har78].

Definition 2.1. *A context-free grammar $G = (N, T, P, S)$ is in 2-Greibach Normalform (2-GNF), iff the set P contains only rules of the form*

$$
\begin{array}{llll}
A & \rightarrow & a & a \in T, \\
A & \rightarrow & aB_1 & a \in T,\ B_1 \in N \backslash \{S\}, \\
A & \rightarrow & aB_1B_2 & a \in T,\ B_1, B_2 \in N \backslash \{S\},\ \textit{and} \\
S & \rightarrow & \lambda. &
\end{array}
$$

Using N. Blum's and R. Koch's transformation method presented in [KB97, BK99], we know that every cfg G can be transformed in $O(|G|^3)$ into a 2-GNF G' so that $L(G) = L(G')$ holds. A number of parsing properties of G' have been studied by [Sto98].

2.2 Semi-Dyck Languages

The following definitions and notations are taken from [Har78], if not otherwise indicated.

An algebraic characterisation of any cfl is obtained on the semi-Dyck set, which is a particular deterministic context-free language[1] (dcfl). The terminal alphabet of D_r, the semi-Dyck set on r-letters, is $(T_r \cup \overline{T_r})$, where $T_r = \{a_1, \ldots, a_r\}$ and $\overline{T_r} = \{\overline{a_i} | a_i \in T_r\}$. We define $T = (T_r \cup \overline{T_r})$ and consider T^* which is a free (word-)monoid with λ as an empty word. We may think of a_i's, $1 \leq i \leq r$, as different kinds of left parenthesis, and $\overline{a_i}$'s, $1 \leq i \leq r$, as their corresponding right parenthesis.

Relations on T^* can be defined using $\rho_0 = \{(a_i\overline{a_i}, \lambda) | a_i \in T_r\} \cup \{(\lambda, a_i\overline{a_i}) | a_i \in T_r\}$. Let $\cong$ be the two-sided congruence relation that contains ρ_0. Not only is $\cong$ an equivalence relation on T^* but $xa_i\overline{a_i}y \cong xy$ for all i, $1 \leq i \leq r$, and all $x, y \in T^*$. Inductively, if $x, y \in T^*$, then $x \cong y$, iff x can be obtained from y by introducing or cancelling adjacent pairs $a_i\overline{a_i}$ or vice versa. A word $w \in T^*$ is reduced for D_r, if w contains no consecutive pairs $a_i\overline{a_i}$ for any i, $1 \leq i \leq r$. Given any word over T^*, we may reduce it by cancelling consecutive pairs. The process of cancelling consecutive pairs can be defined in the following way

Definition 2.2. *Let be $T = (T_r \cup \overline{T_r})$ for some $r \geq 1$. Define a mapping $\mu : T^* \rightarrow T^*$ as follows*

$$\mu(\lambda) = \lambda,$$

$$\mu(xa_i) = \mu(x)a_i \text{ for each } i, 1 \leq i \leq r,$$

$$\mu(x\overline{a_i}) = \begin{cases} \mu(x)\overline{a_i} & \text{if } \mu(x) \notin T^*a_i, \quad 1 \leq i \leq r, \\ x' & \text{if } \mu(x) = x'a_i. \end{cases}$$

The function μ is helpful in dealing with and proving properties of a semi-Dyck set in general. The following properties of μ are inherent. A proof of such listed properties is also given in [Har78]. Let be $T = T_r \cup \overline{T_r}$ for some $r \geq 1$ and let be $u, v, w, x, y \in T^*$.

[1] The deterministic context-free languages are characterized by deterministic pushdown automata which accept them [Sal73, Har78].

(a) $\mu(xa_i\overline{a_i}) = \mu(x)$ for any i, $1 \leq i \leq r$.
(b) $\mu(w)$ is reduced.
(c) $\mu(w) \cong w$.
(d) If u is reduced, $\mu(u) = u$.
(e) $\mu(\mu(w)) = \mu(w)$.
(f) $\mu(xy) = \mu(\mu(x)y)$.
(g) $\mu(xy) = \mu(\mu(x)\mu(y))$.
(h) If $\mu(x) = \mu(y)$, then $\mu(uxv) = \mu(uyv)$.
(i) $\mu(xa_i\overline{a_i}y) = \mu(xy)$ for any i, $1 \leq i \leq r$.
(j) $\mu(x) = \mu(y)$, iff $x \cong y$.
(k) Let be $w \in T^*$. If $y \in T^*$, y is reduced and $y \cong w$, then $y = \mu(w)$.

For generating a semi-Dyck set, we can define a cfg. Let $r \geq 1$ and define $G_r = (N_r, T, P, S)$, where $T = T_r \cup \overline{T_r}$, $N_r = \{S\}$, and the set of rules P is of the form

$$\begin{array}{lcl} S & \to & Sa_iS\overline{a_i}S \\ S & \to & \lambda \end{array}$$

for each i, $1 \leq i \leq r$. Now, the semi-Dyck set is generated by $D_r = L(G_r)$ and the language D_r is defined by $D_r = \{w \in T^* | \mu(w) = \lambda\}$. D_r is a dcfl. Therefore D_r is unambiguous.

Some characteristics of the mapping $\mu : T^* \to T^*$ can be overtaken to D_r. We list them here. Check the proofs as they are presented in [Har78]. Let be D_r a semi-Dyck set.

(a) If $x, y \in D_r$, then $xy \in D_r$.
(b) If $x \in D_r$, then $a_ix\overline{a_i} \in D_r$ for all i, $1 \leq i \leq r$.
(c) For each word $x \in D_r$, either $x = \lambda$ or $x = a_iy\overline{a_i}z$ for some i, $1 \leq i \leq r$, and some $y, z \in D_r$.
(d) If $a_i\overline{a_i}z \in D_r$, then $z \in D_r$.
(e) If $yz \in D_r$ and $y \in D_r$, then $z \in D_r$.
(f) $x \in init(D_r) = \{w \in T^* | wy \in D_r \text{ for some } y \in T^*\}$, iff $\mu(x) \in T_r^*$.

Cfl's can be characterized in terms of D_r.

2.3 The Hardest Context-free Language

There is one cfl L_0 from which any cfl L may be obtained by an inverse homomorphism. That is, for $L \in T^*$, there exists a homomorphism ϕ so that $L = \phi^{-1}(L_0)$. Any cfl can be parsed in whatever time or space it takes to recognize L_0. The definition of L_0 is based on the semi-Dyck set. Do not choose $L_0 = D_2$, where D_2 is deterministic and hence unambiguous. For any homomorphism ϕ, $\phi^{-1}D_2$ is unambiguous, and the generated language is not inherently ambiguous, see Sec. 2.2. We have to choose the nondeterministic version of D_2 for generating L_0 in the following way[2]

Definition 2.3. *Let $T = \{a_1, a_2, \overline{a_1}, \overline{a_2}, \not{c}, c\}$. Define*

$$L_0 = \{\lambda\} \cup \{x_1cy_1cz_1d \ldots x_ncy_ncz_nd | n \geq 1, \quad y_1 \ldots y_n \in \not{c}D_2, x_i, z_i \in T^* \text{ for all } i,\ 1 \leq i \leq n, \\ y_i \in a_1, a_2, \overline{a_1}, \overline{a_2}\}^* \text{ for all } i \geq 2\}$$

[2]This is one of the most important means for coding all cfl using L_0.

Note, that the x_i's and z_i's can contain c's and $\not{c}$'s.

The following theorem is well-known [Gre73]. We use the presentation as given in [Har78].

Theorem 2.1. *If L is a cfl, there is a homomorphism ϕ so that $L = \phi^{-1}(L_0)$ if $\lambda \in L$ and $L = \phi^{-1}(L_0 \backslash \{\lambda\})$ if $\lambda \notin L$.*

3 The Hardest Context-free Language Revised

A key for proving the theorem 2.1 is that there is the transformation of a cfg into GNF [Gre65, Gre73]. A new method for transforming a cfg G into 2-GNF G' in $O(|G|^3)$ has been published in [KB97, BK99]. We will renew the proof using the properties of a cfg in 2-GNF. Of course, this does not lead to a new theorem, but enables us to enlist some new properties for parsing cfl.

3.1 A New Proof for Theorem 2.1

Let be $L \subseteq T^*$ and $L = \phi^{-1}(L_0)$, where ϕ^{-1} is a well-defined inverse homomorphism as in Sec. 2.3. The correctness of the Theorem 2.1 is proved in the following.

Let $\Rightarrow_{lm}$ be a relation – called leftmost derivation – defined by $yA\alpha \Rightarrow_{lm} yw\alpha$, where $A \to w$ is a production in P, $\alpha, w \in (N \cup T)^*$ and $y \in T^*$. Let γ_1, $\gamma_2 \in (N \cup T)^*$ be strings, then $\gamma_1 = yA\alpha$ derives (directly) leftmost in $\gamma_2 = yw\alpha$ in G using $A \to w \in P$ and it is denotated as $\gamma_1 \Rightarrow_{lm} \gamma_2$. We write $\Rightarrow^*_{lm}$ for the reflexive transitive closure of $\Rightarrow_{lm}$.

Let us start the new proof.

Proof There is still no loss of generality, if we consider that $\lambda \notin L$ (if $\lambda \in L$, then it holds $L = \phi^{-1}(L_0)$). We assume, again without the loss of generality, that $L = L(G)$, where $G = (N, T, P, S)$ is in 2-GNF, see Def. 2.1. Thus every rule r_i in P is of the form (if $\lambda \notin L$)

(i)	$A \to aB_1B_2$	$A \in N$, $a \in T$, and $B_1, B_2 \in N \backslash \{S\}$
(ii)	$A \to aB_1$	$A \in N$, $a \in T$, and $B_1 \in \backslash \{S\}$
(iii)	$A \to a$	$A \in N$, and $a \in T$

Let us index the set of nonterminals N as $\{A_1, \ldots, A_n\}$, where $A_1 = S$.

The idea of the construction is to encode rules of the given form of 2-GNF. The encoding includes only nonterminals, not terminals, because a nonterminal would allow to generate a new subtree. Thus, when ϕ is defined on a terminal a, it will encode all possible productions whose righthand side starts with a. Now, we have to define *three mappings*[3] from P into T^*.

(i)	If r_k is	$A_i \to a_i A_{i+1} A_{i+2}$	then $\overline{\tau} r_k = \overline{a_1} a_2^i \overline{a_1} a_1 a_2^{i+1} a_1 a_1 a_2^{i+2} a_1$.
(ii)	If r_k is	$A_i \to a_i A_{i+1}$	then $\overline{\tau} r_k = \overline{a_1} a_2^i \overline{a_1} a_1 a_2^{i+1} a_1$.
(iii)	If r_k is	$A_i \to a_i$	then $\overline{\tau} r_k = \overline{a_1} a_2^i \overline{a_1}$.

[3] instead of two mappings in former times for rules of the form $A \to a\alpha$, where $A \in N$, $a \in T$ and $\alpha \in (N \backslash \{S\})^*$ [Gre65, Har78], because we have three types of rules as given above.

To define τ, we recall that i is the index of the lefthand side and

$$\begin{array}{lll} \underline{if}\ i \neq 1 & \underline{then} & \tau r_k = \overline{\tau} r_k \\ & \underline{else} & \tau r_k = \not{c} a_1 a_2 a_1 \overline{\tau} r_k \end{array}$$

Since τ and $\overline{\tau}$ encode only the nonterminals in the rules and not the terminal, we let $P' = \{r_{p_1}, \ldots, r_{p_m}\}$ be the set of all rules whose righthand side begins with $a \in T$.

We define the homomorphism ϕ by

$$\begin{array}{lll} \phi a = \underline{if} P' \neq \emptyset & \underline{then} & c\tau(r_{p_1}) \ldots c\tau(r_{p_m}) cd \\ & \underline{else} & \not{c}\ \not{c}. \end{array}$$

The following rewritten claim is the key of the proof and gives the exact correspondence between the derivations in G and the structure of L_0.

Claim [= Induction hypothesis] For each $b_1 \ldots b_k \in T_i$, $A_{i_1}, \ldots A_{i_r} \in N \backslash \{S\}$, we have that

$$S \Rightarrow^*_{lm} b_1 \ldots b_k A_{i_1} \ldots A_{i_r}$$

under a sequence of rules $r_{p_1}, \ldots r_{p_k}$, iff there exist x_i, y_i, z_i, $1 \leq i \leq k$ such that

(i) $\phi(b_1 \ldots b_k) = x_1 c y_1 c z_1 d \ldots x_k c y_k c z_k d$

(ii) $y_1 \ldots y_k \in init(\not{c} D_2) = \{x | xw \in \not{c} D_2 \text{ for some } w \in T^*\}$

(iii) $\mu(y_1 \ldots y_k) = \not{c} a_1 a_2^{i_r} a_1 \ldots a_1 a_2^{i_1} a_1$, where for each w, $\mu(w)$ is a unique reduction word that can be obtained from cancellation as it is given in Sec. 2.2.

It is enough to prove this Claim for showing the Theorem, because (ii) and (iii) hold, iff $y_1 \ldots y_k \in \not{c} D_2$; we have that $w \in L$, iff there exists x_i, y_i, z_i, $1 \leq i \leq k$ such that

(i) $\phi w = x_1 c y_1 c z_1 d \ldots x_k c y_k c z_k d$,

(ii) $y_1 \ldots y_k \in \not{c} D_2$, and

(iii) $y_1 = r_{p_i} \in T_2^*$ for $i \geq 2$,

which only holds if $w \in L_0 \backslash \{\lambda\}$.

Note, that using 2-GNF such rules are only of the form $A_i \to aB_1B_2$ or $A_i \to aB_1$ as it is defined in Def. 2.1. The righthand sides of such rules are limited so that we get a new implication for the rules so that they are indexed by $A_i \to a_i A_{i+1} A_{i+2}$ or $A_i \to a_i A_{i+1}$. This means that we can (re)use the index i for describing a corresponce in the next substitutions of the derivation process.

We prove the claim by induction on k.

Induction basis $k = 1$. We have to consider three cases dependently on the form of the rule in 2-GNF.

Case 1 The derivation is $A_1 \Rightarrow a$, where this rule is r_{p_i}.
Then applying τ to this rule gives $\tau r_{p_i} = \not{c} a_1 a_2 a_1 \overline{a_1 a_2 a_1}$. If $P' = \{r_{p_1}, \ldots, r_{p_m}\}$, we have $\phi a = c\tau(r_{p_1}) \ldots c\tau(r_{p_m}) cd = x_1 c y_1 c z_1 d$. If we let $y_1 = \tau(r_{p_i})$ then $\mu(y_1) = \not{c}$ and so $y_1 \in init(\not{c} D)$, and all properties of the claim are fullfilled. Conversely, if each part of the claim is satisfied, there must be a derivation $A_1 \Rightarrow a$ and the Case 1 is verified.

Case 2 The derivation is $A_1 \Rightarrow aA_{1+1} = aA_2$, where $A_2 \in N\backslash\{A_1\}$, and the rule is r_{p_i}. By definition $\tau r_{p_i} = \not{c}a_1a_2a_1\overline{a_1a_2a_1}a_1a_2^ia_1$. If $P' = \{r_{p_1},\ldots,r_{p_m}\}$ then $\phi a = c\tau(r_{p_1})\ldots c\tau(r_{p_m})cd = x_1cy_1z_1d$ satisfying property (i). As in (i), we let $y_1 = \tau(r_{p_i})$. Then $\mu(y_1) = \not{c}a_1a_2^ia_1$ satisfying property (iii). Surely, it holds $y_1 \in init(\not{c}D)$. Conversely, if each part of the claim is fullfilled, there must be a derivation $A_1 \Rightarrow aA_2$.

Case 3 The derivation is $A_1 \Rightarrow aA_{1+1}A_{1+2} = aA_2A_3$, where $A_2,A_3 \in N\backslash\{A_1\}$, and the rule is r_{p_i}. By definition $\tau r_{p_i} = \not{c}a_1a_2a_1\overline{a_1a_2a_1}a_1a_2^ia_1a_1a_2^{i+1}a_1$. If $P' = \{r_{p_1}\ldots r_{p_m}\}$, then $\phi a = c\tau(r_{p_1})\ldots c\tau(r_{p_m})cd = x_1cy_1cz_1d$ satisfying property (i) of the claim. As required in (i), we let $y_1 = \tau(r_{p_i})$. Then $\mu(y_1) = \not{c}a_1a_2^ia_1a_1a_2^{i+1}a_1$ satisfying property (i). Conversely, $y_1 \in init(\not{c}D)$, satisfying property (ii). And if each part of the Claim is satisfied, there must be a derivation $A_1 \Rightarrow aA_2A_3$.

Induction step (one direction of the proof) Assume the result for $k \geq 1$. Suppose we have $S \Rightarrow_{lm}^* b_1\ldots b_kA_{i_1}\ldots A_{i_r}$ and let $\phi(b_1\ldots b_k) = x_1cy_1cz_1d\ldots x_kcy_kcz_kd$, where $\mu(y_1\ldots y_k) = \not{c}a_1a_1^{i_r}a_1\ldots a_1a_2^{i_1}a_1$.

Case 1 Firstly, we check substitution only in terminals b_{k+1}. Let $y_{k+1} = \tau(r_p) = \overline{a_1}\overline{a_2^{i_1}}\overline{a_1}$ so that r_p corresponds to $A_{i_1} \to b_{k+1}$. Then it holds $\phi(b_{k+1}) = x_{k+1}cy_{k+1}cz_{k+1}d$.
We compute stepwise $\mu(y_1\ldots y_ky_{k+1}) = \mu(y_1\ldots y_k)\mu(y_{k+1})$, see Sec. 2.2. So it is $\mu(y_1\ldots y_k)\mu(y_{k+1}) = \mu(a_1a_2^{i_r}a_1\ldots a_1a_2^{i_1}a_1\overline{a_1}\overline{a_2^{i_1}}\overline{a_1}) = \not{c}a_1a_2^{i_r}\ldots a_1a_2^{i_2}a_1$, while $S \Rightarrow_{lm}^* b_1\ldots b_kb_{k+1}A_{i_2}\ldots A_{i_r}$, where $A_{i_1} \to b_{k+1}$.

Case 2 Consider rules of the form $A_i \to b_iA_{i+1}$. Let $y_{k+1} = \tau(r_p) = \overline{a_1}\overline{a_2^{i_1}}\overline{a_1}a_1a_2^{i_{1+1}}a_1$ so that r_p corresponds to $A_{i_1} \to b_{k+1}A_{i_{1+1}}$. Therefore is $\phi(b_{k+1}) = x_{k+1}cy_{k+1}cz_{k+1}d$. Apply $\mu(y_1\ldots y_ky_{k+1}) = \mu(y_1\ldots y_k)\mu(y_{k+1})$, cmp. Sec. 2.2, to $\mu(y_1\ldots y_k)\mu(y_{k+1}) = \mu(a_1a_2^{i_r}a_1\ldots a_1a_2^{i_1}a_1\overline{a_1}\overline{a_2^{i_1}}\overline{a_1}a_1a_2^{i_{1+1}}a_1) = \not{c}a_1a_2^{i_r}a_1\ldots a_1a_2^{i_2}a_1$, while $S \Rightarrow_{lm}^* b_1\ldots b_kb_{k+1}A_{i_2}\ldots A_{i_r}$, where $A_{i_1} \to b_{k+1}A_{i_2}$.

Case 3 Now, we consider rules of the form $A_i \to b_iA_{i+1}A_{i+2}$. Let $y_{k+1} = \tau(r_p) = \overline{a_1}\overline{a_2^{i_1}}\overline{a_1}a_1a_2^{i_1+1}a_1a_1a_2^{i_1+2}a_1$ so that r_p corresponds to $A_{i_1} \to b_{k+1}A_{i_1+1}A_{i_1+2}$. Reasonably, it is $\phi(b_{k+1}) = x_{k+1}cy_{k+1}cz_{k+1}d$. By computing $\mu(y_1\ldots y_ky_{k+1}) = \mu(y_1\ldots y_k)\mu(y_k)$, Sec. 2.2, we receive $\mu(y_1\ldots y_k)\mu(y_{k+1}) = \mu(a_1a_2^{i_r}a_1\ldots a_1a_2^{i_1}a_1\overline{a_1}\overline{a_2^{i_1}}\overline{a_1}a_1a_2^{i_{1+1}}a_1a_1a_2^{i_{1+2}}a_1) = \not{c}a_1a_2^{i_r}a_1\ldots a_1a_2^{i_2}a_1a_1a_2^{i_3}a_1$, while $S \Rightarrow_{lm}^* b_1\ldots b_kb_{k+1}A_{i_2}\ldots A_{i_r}$, where $A_{i_1} \to b_{k+1}A_{i_2}A_{i_3}$.

On the other hand it is defined $\phi(b_1\ldots b_k) = x_1cy_1cz_1d\ldots x_{k+1}cy_{k+1}cz_{k+1}d$ and it holds $\mu(y_1\ldots y_{k+1}) \in \not{c}\{a_1,a_2\}^*$. By the construction of ϕ the only way in the Cases 1 to 3 for this to happen if it is $\mu(y_1\ldots y_k) = \not{c}a_1a_2^{i_r}\ldots a_1a_2^{i_1}a_1$ and if we check the following property for the three cases independently.

Case 1 It holds $\mu(y_{k+1}) = \tau(r_p) = \overline{a_1}\overline{a_2^s}\overline{a_1}$, where r_p must be a rule $A_s \to b_{k+1}$.

Case 2 In this case it is true that $\mu(y_{k+1}) = \tau(r_p) = \overline{a_1}\overline{a_2^s}\overline{a_1}a_1a_1^{k+1+1}a_1 = \overline{a_1}\overline{a_2^s}\overline{a_1}a_1^{k+2}$, where r_p must be a rule $A_s \to b_{k+1}A_{k+2}$.

Case 3 We can construct
$\mu(y_{k+1}) = \tau(r_p) = \overline{a_1}\overline{a_2^s}\overline{a_1}a_1a_1^{k+1+1}a_1a_1a_1^{k+1+2}a_1 = \overline{a_1}\overline{a_2^s}\overline{a_1}a_1a_1^{k+2}a_1a_1a_1^{k+3}$, where r_p must be a rule $A_s \to b_{k+1}A_{k+2}A_{k+3}$.

The following fact is correct for all studied cases. But since $\mu(y_1 \dots y_{k+1}) \in\not\{a_1, a_2\}^*$, there must be some cancellation between $\mu(y_1 \dots y_k)$ and $\mu(y_{k+1})$ that is $s = i_1$.

By induction hypothesis $S \Rightarrow_{lm}^* b_1 \dots b_k A_{i_1} \dots A_{i_r}$ and by using the properties of semi-Dyck sets – as they are introduced in Sec. 2.2 – we have

in Case 1 $S \Rightarrow_{lm}^* b_1 \dots b_k b_{k+1} A_{i_2} \dots A_{i_r}$,

in Case 2 $S \Rightarrow_{lm}^* b_1 \dots b_k b_{k+1} A_{k+2} A_{i_2} \dots A_{i_r}$, and

in Case 3 $S \Rightarrow_{lm}^* b_1 \dots b_k b_{k+1} A_{k+2} A_{k+3} A_{i_2} \dots A_{i_r}$.

Therefore the induction has been extended.

□

4 Observations and a New Theorem

In this Sec. 4 we study the implications inherent of the new proof. We list some observations before formalizing these in Cor.'s 4.1 and 4.2. Our main result, Theorem 4.1, follows in Subsec. 4.2.

4.1 Our List of Observations

Obervation 1 Note, that using 2-GNF such rules are only of the form $A \to aB_1B_2$ or $A \to aB_1$ as it is defined in Def. 2.1. The righthand sides of such rules are limited so that we get a new implication for the rules so that the corresponding productions are indexed by $A_i \to a_i A_{i+1} A_{i+2}$ or $A_i \to a_i A_{i+1}$. That means that we can (re)use the index i for describing a corresponce in the next substitutions of the derivation process. Additionally, we get an information about the width of the next possible subtree. It is also limited by maximal $|a_i A_{i+1} A_{i+2}|$ so that it is $\|G\| = 3$.

Observation 2 In the induction step we studied $S \Rightarrow_{lm}^* b_1 \dots b_k A_{i_1} \dots A_{i_r}$. The length of sentential forms are limited, see Observation 1. If s_k is the sentential form $b_1 \dots b_k A_{i_1} \dots A_{i_r}$, we receive the studied three cases again. We get a new interesting detail.

Case 1 is $b_1 \dots b_k A_{i_1} \dots A_{i_r} \Rightarrow_{lm} b_1 \dots b_k b_{k+1} A_{k+1} A_{k+2} A_{i_2} \dots A_{i_r}$, where $A_{k+1} \to b_{k+1} A_{k+2} A_{k+3}$.

Case 2 is $b_1 \dots b_k A_{i_1} \dots A_{i_r} \Rightarrow_{lm} b_1 \dots b_k b_{k+1} A_{k+2} A_{i_2} \dots A_{i_r}$, where $A_{k+1} \to b_{k+1} A_{k+2}$.

Case 3 is $b_1 \dots b_k A_{i_1} \dots A_{i_r} \Rightarrow_{lm} b_1 \dots b_k b_{k+1} A_{i+2} \dots A_{i^r}$, where $A_{k+1} \to b_{k+1}$.

so that the length of the next sentential forms are

in Case 1 $|s_k| = k + r$ and $|s_{k+1}| = k + 1 + r - 1 + 2 = k + r + 2$ and we receive the equation $|s_k| = |s_{k+1}| + 2$;

in Case 2 $|s_k| = k + r$ and $|s_{k+1}| = k + 1 + r - 1 + 1 = k + r + 1$ and it holds $|s_k| = |s_{k+1}| + 1$; and

in Case 3 $|s_k| = k + r$ and $|s_{k+1}| = k + 1 + r - 1 = k + r$ and it is $|s_k| = |s_{k+1}|$.

Observation 3 If we check the worst case of a longest (sub)sentential form using only rules of the form $A_i \to a_iA_{i+1}A_{i+2}$, we receive $A_i \Rightarrow^*_{lm} a_ia_{i+1}\ldots a_{i+k}A_{i+k+1}A_{i+k+2}A_{i+k-1}\ldots A_{i+2} := \alpha_{i+k+1}$. This means that the generated word can not be shorter than $|w| \geq i+2k-1$, where k, $k \geq 1$. Note, that k corresponds to the actual position in the recognized (sub-)word so that k divides the word in the generated part of terminals $a_ia_{i+1}a_{i+2}\ldots a_{i+k}$ and its unprocessed part of nonterminals $A_{i+k+1}A_{i+k+2}A_{i+k-1}\ldots A_{i+2}$. The length of the (sub)word is $i+k$. The unprocessed part has the length $k-1$.

Observation 4 For generating a terminal string of the length i, $i \geq 1$, over a cfg in 2-GNF, we need at least i nonterminals, because each rule in P starts with a terminal $a \in T$ (if $\lambda \notin L(G)$). So if we look at a (sub)sentential form that is produced by applying only rules of the form $A_i \to aA_{i+1}$, we get $A_i \Rightarrow^*_{lm} a_ia_{i+1}a_{i+2}\ldots a_{i+k}A_{i+k+1} := \alpha_{i+k+1}$. There is a lower bound of a length of a new (sub)word that is still to process. This bound is $i+k+1$ (if it is $\lambda \notin L(G)$).

Observation 5 Combining the observations 3 and 4, as given above, we consider that a sequence of derivations with containing nonterminals α_{i+k+1} is bounded. The size of $|\alpha_{i+k}|$ is $i+k+1 \leq |\alpha_{i+k+1}| \leq i+2k-1$ for each i, $1 \leq i \leq n$, $1 \leq k \leq n$, and $i \leq k$.

Observation 6 If $i+k+1 \leq |\alpha_{i+k}| \leq i+2k-1$ then we can obtain that i terminals are already generated, see also Observations 3 and 4. So it holds $k+1 \leq |\alpha_{k+1}| \leq 2k-1$. We consider two new bounds (for $k+1-1=k$). It is $k \leq |\alpha_k| \leq 2(k-1)-1 = 2k-3$ for $k \geq 2$.

The following Cor.'s 4.1 and 4.2 are directly derivated form the listed observations above.

Corollary 4.1. *Let L_0 be the Hardest Context-free Language as defined in Def. 2.3 and L a cfl generated by a cfg $G = (N,T,P,S)$ in 2-GNF so that $L = L(G)$. By encoding G in L_0 the length of a pair of sentential forms $|s_k|$ and $|s_{k+1}|$ $k \geq 1$, for generating each block "$x_icy_icz_id$", $1 \leq i \leq n$, has an upper bound $|s_{k+1}| \leq |s_k|+2$, where $S := s_1$ and $|S| = 1$.*

Proof The statement follows by Observations 2, 3 and 6 as given above.

□

Corollary 4.2. *Let L_0 be the Hardest Context-free Language as defined in Def. 2.3 and L ($\lambda \notin L$) a cfl generated by a cfg $G = (N,T,P,S)$ in 2-GNF so that $L = L(G)$. By encoding G in L_0 the length of a pair of sentential forms $|s_k|$ and $|s_{k+1}|$ $k \geq 1$, for generating each block "$x_icy_icz_id$", $1 \leq i \leq n$, has a lower bound $|s_k| \leq |s_{k+1}|$, where $S := s_1$ and $|S| = 1$.*

Proof The statement follows by Observations 2, 4 and 6 as presented above.

□

4.2 Theorem

Theorem 4.1. *Let L_0 be the Hardest Context-free Language as defined in Def. 2.3 and L ($\lambda \notin L$) a cfl generated by a cfg $G = (N,T,P,S)$ in 2-GNF so that $L = L(G)$. By encoding G in L_0 the length of a pair of sentential forms $|s_k|$ and $|s_{k+1}|$ $k \geq 1$, for generating each block "$x_icy_icz_id$", $1 \leq i \leq n$, is bounded by $|s_k| \leq |s_{k+1}| \leq |s_k|+2$, where $S := s_1$ and $|S| = 1$.*

Proof The statement follows by the Cor.'s 4.1 and 4.2.

□

5 Conclusion

We present a new proof of the existence of the Hardest Context-free Language L_0 in Sec. 3. This given proof is inherent in that the length of a pair of sentential forms $|s_k|$ and $|s_{k+1}|$ $k \geq 1$, for generating each block "$x_i c y_i c z_i d$", $1 \leq i \leq n$, is bounded by $|s_k| \leq |s_{k+1}| \leq |s_k| + 2$, where $S := s_1$ and $|S| = 1$, see Theorem 4.1.

Observe that some other well-known results could be influenced. It is known that a cfg in GNF guarantees that each nonterminal, which generates a terminal string, cannot generate a terminal word of the length less than once. This observation is inherent of any definition of a GNF comparing the traditional GNF transformation [Gre65] or the new method for transforming a cfg in GNF [KB97, BK99] and implies that we can prune any node whose sentential form has the length greater than the given input string w. This fact includes the connection in between the recognition and the parsing of a word w generated over a given cfg G [Woo87]. We will therefore study the impact of Theorem 4.1 on recognizing and parsing of a word generated over a given cfg G.

References

[BK99] Norbert Blum and Robert Koch. Greibach Normal Form Transformation Revisited. *Information and Computation*, 150(1):112–118, 1999.

[Cho59] Noam Chomsky. On Certain Formal Properties of Grammars. *Information and Control*, 2, 1959.

[Gre65] Sheila A. Greibach. A New Normal-Form Theorem for Context-free Phrase Structure Grammars. *JACM*, 12:42–52, January 1965.

[Gre73] Sheila A. Greibach. The hardest context-free language. *SIAM Journal of Computing*, 2:304–310, 1973.

[Har78] Michael A. Harrison. *Introduction to Formal Language Theory*. Addison Wesley, Reading, Massachusetts, 1978.

[KB97] Robert Koch and Norbert Blum. Greibach Normal Form Transformation, Revisited. *STACS 1997*, pages 47–54, 1997.

[Sal73] Arto Salomaa. *Formal Languages*. Academic Press, New York, 1973.

[Sto98] Julia Stoll. Parsing-Eigenschaften effizient zu ermittelnder Greibach Normalformen. Diplomarbeit, Technische Universität Darmstadt, Fachbereich Informatik, Institut für Theoretische Informatik, Fachgebiet Automatentheorie und Formale Sprachen, 1998.

[Woo87] Derick Wood. *Theory of Computation*. John Wiley & Sons, 1987.