Two Characterizations of Regularity of Complete CF(¢,$)-Grammars

Martin Plátek[1], František Mráz[1,*], Dana Pardubská[2] and Marta Vomlelová[1]

[1]Charles University, Department of Computer Science, Malostranské nám. 25, 118 00 Praha 1, Czech Republic
[2]Comenius University in Bratislava, Department of Computer Science, Mlynská Dolina, 84248 Bratislava, Slovakia

Abstract

The class of context-free grammars is the only class of languages in the Chomsky hierarchy that is not closed under complement. A complete CF(¢,$)-grammar is an extended context-free grammar that generates both a language and its complement. Hence, complete CF (¢,$)-grammars enable the study of the class of context-free languages that are closed under complement. We introduce and study two types of reductions based on complete CF(¢, $)-grammars and their languages. Both types of reduction used in this paper ensure correctness and error-preserving analysis by reducing each word over its input alphabet. We present two relaxed models of complete CF(¢,$)-grammars that characterize regularity, and we show hierarchies of grammars and separation results for these models in terms of complete CF(¢,$)-grammars.

Keywords

context-free language, regular language, context-free grammar, complete CF(¢,$)-grammar

1. Introduction

This paper is a continuation of the papers [1, 2], and is also inspired by [3, 4]. Paper [1] focuses on non-regularity and pumping tests with the aim of identifying conditions under which the languages generated by a complete CF(¢, $)-grammar are non-regular. Paper [2] shifts from non-regularity tests to regularity tests. It investigates restrictions guaranteeing that a complete CF(¢, $)-grammar generates only regular languages.

Here, we study complete context-free grammars with sentinels and two types of reductions of their words that are separated by a prefix and a suffix. The first type is described as a property of their languages, while the second one is described as a property of grammars. These notions are motivated by the linguistic method called analysis by reduction; see [5, 6, 7]. Namely, they can be useful for technical support for the normalization of natural (national) and artificial languages.

Linguistic analysis by reduction serves as a tool for constructing formal grammars from linguistic observations. Analysis by reduction checks the (non)correctness of an input sentence by stepwise rewriting some part(s) of the current form with a reduced (shorter) one until we obtain a simple sentence for which we can easily decide its correctness. In general, analysis by reduction is nondeterministic. One step of the analysis consists of rewriting a substring of length bounded by a constant with a shorter string. An input sentence is accepted if there is a sequence of reductions that ends with a final simple sentence from the language. Then, of course, all intermediate sentences obtained during the analysis are also accepted. Each reduction must be *error-preserving*; that is, no sentence outside the target language can be rewritten into a sentence of the target language.

Here, we focus on two restricted versions of the analysis by reduction. At first, we require that at each step of the analysis by reduction, the current word is not rewritten. Instead, a continuous segment of the current word is deleted. Second, we consider the reduction analysis for languages generated only by so-called complete grammars.

ITAT'26: Information Technologies – Applications and Theory, September 25–29, 2026, Biele Karpaty, Slovakia
*Corresponding author.
martin.platek@mff.cuni.cz (M. Plátek); frantisek.mraz@mff.cuni.cz (F. Mráz); pardubska@dcs.fmph.uniba.sk (D. Pardubská); marta.vomlelova@mff.cuni.cz (M. Vomlelová)
0000-0003-3147-6442 (M. Plátek); 0000-0001-3869-3340 (F. Mráz); 0000-0001-9383-8117 (D. Pardubská); 0000-0001-9104-804X (M. Vomlelová)
© 2026 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

CEUR Workshop Proceedings ceur-ws.org ISSN 1613-0073
published 2026-11-09

CEUR-WS.org/Vol-4323/paper19.pdf

Informally, a complete grammar (with sentinels ¢ and $) G_C is an extended context-free grammar with two initial nonterminals S_A and S_R. This grammar has a finite alphabet Σ of terminals not containing ¢ and $, a finite alphabet of nonterminals, and a set of context-free rewriting rules of the form $X \to \alpha$, where X is a nonterminal and α is a string of terminals, nonterminals, and sentinels ¢, $. The language generated by the grammar is the set $\{¢\} \cdot \Sigma^* \cdot \{\$\}$. The set of words generated from the initial nonterminal S_A, called the accepting language, is a language of the form $\{¢\} \cdot L \cdot \{\$\}$, where $L \subseteq \Sigma^*$. Analogously, the set of words generated from the second initial nonterminal S_R, called the rejecting language, is exactly $\{¢\} \cdot (\Sigma^* \setminus L) \cdot \{\$\}$.

The paper is structured as follows. Section 2 introduces the basic notions. Section 3 introduces PSFL(c)-grammars, shows that the languages they generate are regular, and presents a hierarchy of PSFL(c)-grammars based on the parameter c. Section 4 introduces PSFG(d)-grammars, where d is a positive integer, and shows that the class of PSFG-grammars is a subclass of the class of PSFL-grammars. We show that there is a PSFL(4)-grammar that is not a PSFG(d)-grammar for any natural number d. Further, we present additional separation results. The last section presents conclusions and ideas for future work.

2. Basic notions

Definition 1 (CF(¢,$)-grammars). *Let N and Σ be two disjoint alphabets, $¢, \$ \notin (N \cup \Sigma)$ and $G = (N, \Sigma \cup \{¢, \$\}, S, P)$ be a context-free grammar generating a language $L \subseteq \Sigma^*$, of the form $\{¢\} \cdot L \cdot \{\$\}$, where N is a finite set of nonterminals, $\Sigma \cup \{¢, \$\}$ is a set of terminals, S is the initial nonterminal and S does not occur in the right-hand side of any rule (production) in a finite set of rules P, each rule in P is of the form $X \to \alpha$, where $X \in N$ and $\alpha \in (N \cup \Sigma \cup \{¢, \$\})^*$. We say that G is a CF(¢,$)-grammar. The language L is the* internal language of G, *and is denoted as $L_{\text{in}}(G)$.*

The right sentinel $ in CF(¢,$)-grammars facilitates language recognition. While each deterministic context-free language L can be generated by an LR(1) grammar, both $L \cdot \{\$\}$ and $\{¢\} \cdot L \cdot \{\$\}$ are generated by even simpler LR(0) grammars [8]. The left sentinel ¢ is included in CF(¢,$)-grammars for compatibility with restarting automata [9]. The closure properties of context-free languages [10] imply that the class of internal languages of CF(¢,$)-grammars characterizes the class of context-free languages (CFL).

Definition 2. *Let $G_C = (N, \Sigma \cup \{¢, \$\}, S, P)$ be a CF(¢,$)-grammar. Then G_C is called a* complete CF(¢,$)-grammar *if*

1. *$S \to S_A \mid S_R$, where $S_A, S_R \in N$, are the only rules in P that contain the initial nonterminal S. No other rule of G_C contains S_A or S_R on its right-hand side.*
2. *The languages $L(G_A)$ and $L(G_R)$ generated by the grammars $G_A = (N, \Sigma \cup \{¢, \$\}, S_A, R)$ and $G_R = (N, \Sigma \cup \{¢, \$\}, S_R, R)$, respectively, are disjoint and complementary with respect to $\{¢\} \cdot \Sigma^* \cdot \{\$\}$. That is, $L(G_A) \cap L(G_R) = \emptyset$ and $L(G_C) = L(G_A) \cup L(G_R) = \{¢\} \cdot \Sigma^* \cdot \{\$\}$.*

We will denote the complete CF(¢,$)-grammar G_C as a pair $G_C = (G_A, G_R)$. Furthermore, we will denote the accepting and rejecting grammars of the complete CF(¢,$)-grammar G_C by G_A and G_R, respectively.

For each word of the form $¢w\$$, where $w \in \Sigma^*$, there exists some derivation tree T according to G_C. The node under the root of T is labeled either S_A or S_R. If it is S_A, the word is generated by the accepting grammar G_A. Otherwise, it is generated by the rejecting grammar G_R.

Realize that although two or more derivation trees can exist for each word, either all of them are accepting, or all of them are rejecting.

3. PSFL(c)-grammars

In this section, we transform the regularity tests based on pumping properties from [2] into restricted types of reductions. Examples illustrating the following notions can be found in Sections 3.3 and 4.1 below.

In what follows, we introduce, in two steps, a class of complete CF(¢, $)-grammars, which guarantee that both their accepting and rejecting languages are regular. In this section, we use only restrictions based on the properties of $L(G_A)$ and $L(G_R)$. On the other hand, we will later in this paper compare the class of grammars introduced here with a class of grammars introduced later. We stress that we consider our results as results on the comparison of (sub)classes of context-free grammars.

Next, we will define complete CF(¢, $)-grammars that enable correctness-preserving reductions in each sufficiently long prefix and each sufficiently long suffix of any word.

Definition 3. *Let $G_C = (G_A, G_R) = (N, \Sigma \cup \{¢, \$\}, S, P)$ be a complete CF(¢, $)-grammar, $c > 0$ be a constant, and x, s, x_1, u_1 be words over $\Sigma \cup \{¢, \$\}$ such that the following conditions are fulfilled:*

- *(Ap) for each $xs \in L(G_A)$ such that $|x| > c$ we can write $x = x_1u_1y_1$ such that $0 < |u_1| \leq c$, and $x_1y_1s \in L(G_A)$.*
- *(As) for each $xs \in L(G_A)$ such that $|s| > c$ we can write $s = x_1u_1y_1$ such that $0 < |u_1| \leq c$, and $xx_1y_1 \in L(G_A)$.*
- *(Rp) for each $xs \in L(G_R)$ such that $|x| > c$ we can write $x = x_1u_1y_1$ such that $0 < |u_1| \leq c$, and $x_1y_1s \in L(G_R)$.*
- *(Rs) for each $xs \in L(G_R)$ such that $|s| > c$ we can write $s = x_1u_1y_1$ such that $0 < |u_1| \leq c$, and $xx_1y_1 \in L(G_R)$.*

In conditions (Ap) and (Rp), the suffix s can be an empty word. Similarly, in conditions (As) and (Rs), the prefix x can be an empty word.

We say that G_C is a PSFL1(c)-grammar. The notation arises from the fact that in Prefixes and in Suffixes we require Free reducing in languages generated by PSFL1(c)-grammars.

Note that the previous definition is based on the reducing properties of $L(G_A)$ and $L(G_R)$ only; no structural properties of the complete CF(¢,$)-grammar G_C are explicitly assumed, and no pumping is explicitly assumed. This is a crucial difference from our attempt in [2], where we required that the omitted string can be pumped any number of times (including zero, as used in Definition 3). We will show that our current approach is more general than the former.

If G_C is a PSFL1(c)-grammar, then each long enough word $w_1w_2 \in L(G_C)$ contains a prefix (suffix) subword after the deletion of which the resulting word belongs to $L(G_x) \Leftrightarrow w_1w_2 \in L(G_x)$, $x \in \{A, R\}$. In order to simplify the reduction process, we shall consider the "prefix/suffix" and "leftmost/rightmost" reductions of w_1w_2 defined below.

Definition 4 (PS-cut xs, P-reductions, S-reductions). *Let $G_C = (G_A, G_R) = (N, \Sigma \cup \{¢, \$\}, S, P)$ be a complete CF(¢, $)-grammar, $c > 0$ be a constant, and let x, s, x_1, y_1, be words.*

- *(ps) Let $xs \in L(G_C)$. We say that xs is a prefix-suffix cut by G_C (PS-cut by G_C, PS-cut if G_C is known from the context). The division between the prefix x and the suffix s will be known from the context.*
- *(ap) Let $xs \in L(G_A)$ fulfills the conditions (Ap).*
  We write $xs \rightsquigarrow_{P,G_C} x_1y_1s$ and say that $xs \rightsquigarrow_{P,G_C} x_1y_1s$ is a P-reduction of xs by G_C, c (a c-reduction in the Prefix).
- *(as) Let $xs \in L(G_A)$ fulfills the conditions (As).*
  We write $xs \rightsquigarrow_{S,G_C} xx_1y_1$ and say that $xs \rightsquigarrow_{S,G_C} xx_1y_1$ is an S-reduction of xs by G_C, c (a c-reduction in the Suffix).

(rp) *Let $xs \in L(G_R)$ fulfills the conditions (Rp).*
We write $xs \rightsquigarrow_{P,G_C} x_1y_1s$ and say that $xs \rightsquigarrow_{P,G_C} x_1y_1s$ is a P-reduction of xs by G_C, c (a c-reduction in the Prefix).

(rs) *Let $xs \in L(G_R)$ fulfills the conditions (Rs).*
We write $xs \rightsquigarrow_{S,G_C} xx_1y_1$ and say that $xs \rightsquigarrow_{S,G_C} xx_1y_1$ is an S-reduction by G_C, c of xs (a c-reduction in the Suffix).

The relations $\rightsquigarrow^*_{S,G_C}$ and $\rightsquigarrow^*_{P,G_C}$ are the reflexive and transitive closure of $\rightsquigarrow^*_{S,G_C}$ and $\rightsquigarrow^*_{P,G_C}$, respectively.

Definition 5 (PSFL(c)-grammars). *Let $G_C = (G_A, G_R)$ be a PSFL1(c)-grammar. We say that G_C is a PSFL(c)-grammar if the following additional condition (A) is fulfilled for all words x, x', s, s', y, y':*

(A) *If $xs' \rightsquigarrow^*_{P,G_C} x's'$, $ys' \rightsquigarrow^*_{P,G_C} x's'$, and $xs \rightsquigarrow^*_{S,G_C} xs'$ then there is a (composed) reduction such that $ys \rightsquigarrow^*_{S,G_C} ys'$.*

Condition (A) in the definition of PSFL(c)-grammar means that if two words xs' and ys' can be reduced to the same word $x's'$ using P-reductions and xs' can be obtained from xs using S-reductions, then the same S-reductions can be applied to ys to obtain ys'. Hence, the reductions in prefixes are independent of the reductions in suffixes.

3.1. Technical notions

To show the regularity of accepting languages defined by PSFL(c) grammars, we utilize the characterization of regular languages by the right congruence of a finite index (Theorem 3.1, [11]). The idea is to construct classes of words, where one class contains all words that can be reduced to the same shortest irreducible word by applying P- and S-reductions. The following technical notions serve to construct the desired relation gradually.

Definition 6 (Pl-reductions, Sr-reductions). *Let $G_C = (G_A, G_R)$ be a PSFL(c)-grammar, and x, x_1, u_1, y_1, s be words.*

(apl) *Let $xs \in L(G_A)$, $x = x_1u_1y_1$, and $xs \rightsquigarrow_{P,G_C} x_1y_1s$ be a P-reduction of xs by G_C be such that x_1u_1 is the shortest prefix such that $x_1y_1s \in L(G_A)$; if there are more possibilities to divide the prefix $x_1u_1 = x_1'u_1'$ so that $x_1'y_1s \in L(G_A)$, we take x_1u_1 for which x_1 is the shortest possible.*
We write $xs \rightsquigarrow_{Pl,G_C} x_1y_1s$ and say that $xs \rightsquigarrow_{Pl,G_C} x_1y_1s$ is a Pl-reduction of xs by G_C (the leftmost reduction in Prefix).

(asr) *Let $xs \in L(G_A)$, $s = x_1u_1y_1$, and $xs \rightsquigarrow_{P,G_C} xx_1y_1$ be an S-reduction of xs by G_C such that u_1y_1 is the shortest suffix such that $xx_1y_1 \in L(G_A)$; if there are more possibilities to divide the suffix $u_1y_1 = u_1'y_1'$ so that $xx_1y_1' \in L(G_A)$, we take u_1y_1 for which y_1 is the shortest.*
We write $xs \rightsquigarrow_{Sr,G_C} xx_1y_1$ and say that $x \rightsquigarrow_{Sr,G_C} xx_1y_1$ is an Sr-reduction of xs by G_C (the rightmost reduction in Suffix).

(rpl) *Let $xs \in L(G_R)$, $x = x_1u_1y_1$, and $xs \rightsquigarrow_{P,G_C} x_1y_1s$ be a P-reduction of xs by G_C such that x_1u_1 is the shortest prefix such that $x_1y_1s \in L(G_R)$; if there are more possibilities to divide the prefix $x_1u_1 = x_1'u_1'$ so that $x_1'y_1s \in L(G_R)$, we take x_1u_1 for which x_1 is the shortest.*
We write $xs \rightsquigarrow_{Pl,G_C} x_1y_1s$ and say that $xs \rightsquigarrow_{Pl,G_C} x_1y_1s$ is a Pl-reduction of xs by G_C (the leftmost reduction in the Prefix).

(rsr) *Let $xs \in L(G_R)$, $s = x_1u_1y_1$, and $xs \rightsquigarrow_{P,G_C} xx_1y_1$ be an S-reduction of xs by G_C such that u_1y_1 is the shortest suffix such that $xx_1y_1 \in L(G_R)$; if there are more possibilities to divide $u_1y_1 = u_1'y_1'$ so that $xx_1y_1' \in L(G_R)$, we take u_1y_1 for which y_1 is the shortest.*
We write $xs \rightsquigarrow_{Sr,G_C} xx_1y_1$ and say that $xs \rightsquigarrow_{Sr,G_C} xx_1y_1$ is an Sr-reduction by G_C of xs (the rightmost reduction in Suffix).

The relations $\rightsquigarrow^*_{Sr,G_C}$ and $\rightsquigarrow^*_{Pl,G_C}$ are the reflexive and transitive closure of $\rightsquigarrow_{Sr,G_C}$ and $\rightsquigarrow_{Pl,G_C}$, respectively.

Remark. Note that any Sr-reduction is an S-reduction and any Pl-reduction is a P-reduction.

Basic PS-prefix, PS-suffix by G_C

Let $G_C = (G_A, G_R)$ be a PSFL(c)-grammar, for some $c > 0$, and $xs \in L(G_C)$, for some words x, s.

- If $|x| \leq c$, we say that x is a *basic PS-prefix by* G_C.
- If $|s| \leq c$, we say that s is a *basic PS-suffix by* G_C.

Other technical notions

Let $G_C = (G_A, G_R)$ be a PSFL(c)-grammar, with sentinels ¢, \$ and terminal alphabet Σ.

The set of nonempty PS-prefixes for G_C will be denoted as

$$Pref = \{¢x \mid x \in \Sigma^*\} \cup \{¢x\$ \mid x \in \Sigma^*\}.$$

The set of all nonempty PS-suffixes for G_C will be denoted as

$$Suff = \{s\$ \mid s \in \Sigma^*\} \cup \{¢s\$ \mid s \in \Sigma^*\}.$$

The set of all basic PS-prefixes will be denoted as

$$BPref(c) = \{x \mid x \in Pref, |x| \leq c\}.$$

The set of all basic PS-suffixes will be denoted as

$$BSuff(c) = \{s \mid s \in Suff, \text{ and } |s| \leq c\}.$$

Consider a long enough string $w = xs \in \{¢\} \cdot \Sigma^* \cdot \{\$\}$. As G_C is a PSFL(c) grammar, xs can be reduced to shorter words. Assume that at first w is reduced within the suffix s by the rightmost reductions, so that $w_r = xs'$ is obtained and $|s'| \leq c$. Then, continue with a series of leftmost reductions within x only, resulting in a word $w_{rl} = x's'$, where $|x'| \leq c$. The obtained w_{rl} will help us identify the right congruence class to which the original w belongs.

In what follows, for a set X, $\mathcal{P}(X)$ denotes the set of all its subsets.

Definition 7 (Prefix-BSuffix characteristic function). *Let $G_C = (G_A, G_R)$ be a PSFL(c)-grammar for some $c > 0$. A characteristic prefix-BSuffix function by G_C is the function*

$$ChpB(G_C) : Pref(G_C) \times BSuff(G_C) \to BPref(G_C)$$

that assigns to each PS-prefix x and basic suffix s_b the basic PS-prefix x_1 to which the PS-prefix x can be reduced when we reduce xs_b by the leftmost Pl-reductions. More formally:

$$ChpB(G_C)(x, s_b) = x_1, \text{ where } \quad x_1 \in BPref(c), xs_b \rightsquigarrow^*_{Pl,G_C} x_1 s_b, \text{ and there is no Pl-reduction for } x_1 s_b \text{ by } G_C.$$

Note that Pl-reductions are deterministic; therefore, there is at most one such x_1.

3.2. Equivalence of PS-prefixes and regularity of the languages generated by PSFL(c)-grammars

Based on the characteristic functions of PS-prefixes, we will define an equivalence on PS-prefixes.

Definition 8. *Let $G_C = (G_A, G_R)$ be a PSFL(c)-grammar. We say that two PS-prefixes x and z are equivalent, and we write $x \cong z$, if*

$$ChpB(G_C)(x, s_b) = ChpB(G_C)(z, s_b) \text{ for all } s_b \in BSuff.$$

Lemma 1. *Let $G_C = (G_A, G_R)$ be a PSFL(c)-grammar, for some positive integer c. The relation $\cong$ defined for pairs of PS-prefixes by G_C is an equivalence relation of a finite index on the set of prefixes $Pref$.*

Proof: It is easy to see that the relation $\cong$ is reflexive, symmetric, and transitive. Thus, it is an equivalence relation. The finite index of $\cong$ follows from the fact that the number of members $BSuff$ is finite (and is determined by G_C and c), and the number of possible values of $ChpB(G_C)$ is finite (and is determined by G_C and c). □

The following lemma is the key property for showing that the accepting and rejecting languages of a PSFL(c)-grammar are regular.

Lemma 2. *Let $G_C = (G_A, G_R)$ be a PSFL(c)-grammar, for some positive integer c, and let $x \cong y$ for some PS-prefixes x, y by G_C. Then, for each PS-suffix s, $xs \in L(G_A) \Leftrightarrow ys \in L(G_A)$.*

Proof: Let x and y be two PS-prefixes by G_C such that $x \cong y$, and let s be a PS-suffix by G_C. Let us suppose $xs \in L(G_A)$. Then $xs \rightsquigarrow^*_{Sr,G_C} xs_b \rightsquigarrow^*_{Pl,G_C} x_b s_b$, for some basic PS-suffix $s_b \in BSuff(c)$ and a basic PS-prefix $x_b \in BPref(c)$.

Then $x \cong y$ implies (Definition 8) $x_b = ChpB(x, s_b) = ChpB(y, s_b)$. Using Condition (A) from the definition of PSFL(c)-grammar (Definition 5), we get $ys \rightsquigarrow^*_{S,G_C} ys_b$. The final step follows from the fact that $x_b s \rightsquigarrow^*_{S,G_C} x_b s_b$ and $ys_b \rightsquigarrow^*_{Pl,G_C} x_b s_b$. As S-reductions and P-reductions are error- and correctness-preserving, all words $x_b s_b$, ys_b, and ys are in $L(G_A)$.

The corresponding statement also holds for $L(G_R)$, which completes the proof that

$$xs \in L(G_A) \Leftrightarrow ys \in L(G_A).$$

□

Important note. Let R_L denote the equivalence with respect to a language L defined in the following way: xR_Ly if and only if for all words z, $xz \in L \Leftrightarrow yz \in L$. According to Theorem 3.1 of [11], the language L is regular if and only if R_L has a finite index.

Lemma 3. *Let $G_C = (G_A, G_R)$ be a PSFL(c)-grammar, for some $c > 0$, and $L = L(G_A) \subseteq (\Sigma \cup \{¢, \$\})^*$. Let R_L denote the equivalence relation on $(\Sigma \cup \{¢, \$\})^*$ defined with xR_Ly if and only if for each word z it holds $xz \in L \Leftrightarrow yz \in L$. Then R_L has a finite index.*

Proof: We will show that the index of R_L is at most the index of $\cong$ increased by 2. Consider the relation R_L where $L = L(G_A)$. We need to show that R_L has a finite index on Γ^*, where $\Gamma = \{¢, \$\} \cup \Sigma$. Obviously,

$$\begin{aligned} \Gamma^* = \; & L(G_A) \quad \cup & (1)\\ & \{x \in \Gamma^* \mid \forall y \in \Gamma^* : xy \notin \{¢\} \cdot \Sigma^* \cdot \{\$\}\} \cup L(G_R) \quad \cup & (2)\\ & \{x \in \Gamma^* \mid \exists y \in \Gamma^+ : xy \in \{¢\} \cdot \Sigma^* \cdot \{\$\}\}. & (3) \end{aligned}$$

Therefore, we will distinguish three cases.

All words in $L = L(G_A)$ (case (1)) are R_L-equivalent, and there are no words outside L that can be R_L-equivalent to a word in L, because the only word we can append to a word from $L(G_A)$ to obtain a word from $L(G_A)$ is the empty word λ.

Similarly, all words in $L' = \{x \in \Gamma^* \mid \forall y \in \Gamma^* : xy \notin \{¢\} \cdot \Sigma^* \cdot \{\$\}\} \cup L(G_R)$ (case(2)) are R_L-equivalent, as appending any word from Γ^* to a word from L' results in a word not in $L = L(G_A)$.

In addition, we will show that $Pref = \{x \in \Gamma^* \mid \exists y \in \Gamma^+ : xy \in \{¢\} \cdot \Sigma^* \cdot \{\$\}\}$, that is, all *proper* prefixes of words from $\{¢\} \cdot \Sigma^* \cdot \{\$\}$ (case (3); note that y is nonempty here) belong to a finite number of different equivalence classes with respect to R_L.

For a contradiction, assume that the words in $Pref$ belong to an infinite number of different classes of equivalence with respect to R_L. Then, for each $n \geq 1$, there exist words $x_1, x_2, \ldots, x_n \in Pref$ that are pairwise R_L-non-equivalent.

If n is greater than the number of equivalence classes with respect to $\cong$ (see Definition 8), then there exist two words x_i, x_j, $x_i \neq x_j$, such that $x_i \cong x_j$, and x_i is not R_L equivalent to x_j. However, each word s is either a PS-suffix or not. If s is a PS-suffix, then according to Lemma 2, $x_i s \in L(G_A) \Leftrightarrow x_j s \in L(G_A)$. If s is not a PS-suffix, then $x_i s \notin L(G_A)$ and $y_j s \notin L(G_A)$. Hence, for all words s, $x_i s \in L(G_A) = L \Leftrightarrow x_j s \in L(G_A) = L$, and $x_i R_L x_j$ – a contradiction to the assumption that x_i is not R_L equivalent to x_j. □

Corollary 1. *Let $G_C = (G_A, G_R)$ be a PSFL(c)-grammar, for some $c > 0$, and $\Gamma = \Sigma \cup \{¢, \$\}$ be the terminal alphabet of G_C. Then there is a right congruence R_L on Γ^* of a finite index such that $L(G_A)$ is a class of the partition of Γ^* given by R_L.*

Theorem 1. *Let $G_C = (G_A, G_R)$ be a PSFL(c)-grammar. Then the languages $L(G_A)$ and $L(G_R)$ are regular.*

Proof: According to the previous corollary, the language $L(G_A)$ is regular. The regularity of the language $L(G_R)$ follows from the closure properties of regular languages. □

Corollary 2. *Let $G_C = (G_A, G_R)$ be a complete CF($¢, \$$)-grammar, and $L(G_A)$ be a non-regular language. Then G_C is not a PSFL(c)-grammar for any natural number c.*

3.3. A hierarchy of PSFL(c)-grammars

Note that PSFL(c) grammars are a special type of complete CF(¢,$)-grammars, implying that the shortest word from $L(G_C)$ is ¢$, which is of length 2. So, if we think of the hierarchy of PSFL(c) grammars with respect to the integer c, the smallest possible class is PSFL(2). As $\{¢\} \cdot \Sigma^* \cdot \{\$\}$ is the accepting language of a PSFL(2)-grammar, this smallest class is not empty.

Proposition 1. *If G_C is a PSFL(c) grammar, then $c \geq 2$.*

Example 1. *Let $G_C^{(c2)} = (\{S, S_A, S_R, B\}, \{a, b, ¢, \$\}, S, R)$ be a complete CF(¢,$)-grammar with the following set of rules:*

$$\begin{array}{lcl} S & \rightarrow & S_A \mid S_R, \\ S_A & \rightarrow & ¢B\$, \\ B & \rightarrow & aB \mid bB \mid \lambda. \end{array}$$

The grammar $G_C^{(c2)}$ is a CF(¢,$)-grammar $G_C^{(c2)} = \left(G_A^{(c2)}, G_R^{(c2)}\right)$, where $G_A^{(c2)}$ has starting symbol S_A, $G_R^{(c2)}$ has starting symbol S_R, and obviously, $L\left(G_C^{(c2)}\right) = L\left(G_A^{(c2)}\right) = \{¢\} \cdot \{a, b\}^* \cdot \{\$\}$, and $L(G_R^{(c2)}) = \emptyset$.

We can see that $G_C^{(c2)} = \left(G_A^{(c2)}, G_R^{(c2)}\right)$ is a PSFL(2)-grammar.

Corollary 3. *There exists a PSFL(2)-grammar.*

Example 2. *Let $G_C^{(c3)} = (\{S, S_A, S_R, B\}, \{a, b, ¢, \$\}, S, R)$ be a complete CF(¢,$)-grammar with the following set of rules:*

$$\begin{array}{lcllcl} S & \rightarrow & S_A \mid S_R, & B & \rightarrow & aB \mid bB \mid a \mid b, \\ S_A & \rightarrow & ¢B\$, & S_R & \rightarrow & ¢\$. \end{array}$$

The grammar $G_C^{(c3)}$ is a complete CF(¢,$)-grammar $G_C^{(c3)} = \left(G_A^{(c3)}, G_R^{(c3)}\right)$, where $G_A^{(c3)}$ has the starting symbol S_A, $G_R^{(c3)}$ has the starting symbol S_R. $L\left(G_A^{(c3)}\right) = \{¢\} \cdot \{a, b\}^+ \cdot \{\$\}$, $L\left(G_R^{(c2)}\right) = \{¢\$\}$, and $L\left(G_C^{(c3)}\right) = L\left(G_A^{(c3)}\right) \cup L\left(G_R^{(c3)}\right) = \{¢\} \cdot \{a, b\}^* \cdot \{\$\}$. It is easy to see that grammar $G_C^{(c3)}$ is a PSFL(3)-grammar, and since ¢a$ cannot be reduced, the grammar is not a PSFL(2)-grammar.

Corollary 4. *There is a PSFL(3)-grammar, which is not a PSFL(2)-grammar.*

Example 3. *Let us consider the complete CF(¢,$)-grammar* $G_C^{(c4)} = \left(G_A^{(c4)}, G_R^{(c4)}\right)$ *that generates the accepting language* $\{¢a^{2n}b^m\$ \mid n > 0, m \geq 0\}$ *with the following set of rules:*

$$\begin{array}{lcllcl} S & \to & S_A \mid S_R, & S_R & \to & ¢A_s\$ \mid ¢B_s\$ \mid ¢\$ \mid ¢a\$, \\ S_A & \to & ¢AB\$, & A_s & \to & aaA_s \mid AbXaX \mid a \mid abX, \\ A & \to & a^2A \mid a^2, & X & \to & aX \mid bX \mid \lambda. \\ B & \to & Bb \mid \lambda, & & & \end{array}$$

It is easy to check that $G_C^{(c4)}$ *is a PSFL(4)-grammar.*

We can also see that the word $¢aa\$$ *is a word of length* 4 *from* $L\left(G_A^{(c4)}\right)$*, and that it cannot be reduced by any reduction of* $G_C^{(c4)}$*. This means that* $G_C^{(c4)}$ *is not a PSFL(3)-grammar.*

The next corollary is a consequence of the previous example.

Corollary 5. *There is a PSFL(4)-grammar, which is not a PSFL(3)-grammar.*

Example 4. *Let* k *be a natural number such that* $k > 1$*. Let us consider the complete CF(¢,$)-grammar* $G_C^{(ck)} = \left(G_A^{(ck)}, G_R^{(ck)}\right)$ *that generates the accepting language* $\{¢a^{k\cdot n}b^m \mid n > 0, m \geq 0\}$ *with the following set of rules:*

$$\begin{array}{lcllcl} S & \to & S_A \mid S_R, & S_R & \to & ¢A_s\$ \mid ¢B_s\$ \mid ¢a^i\$, \text{ where } 1 \leq i \leq k-1, \\ S_A & \to & ¢AB\$, & A_s & \to & a^kA_s \mid AbXaX \mid a^i \mid a^ibX, \text{ where } 1 \leq i \leq k-1, \\ A & \to & a^kA \mid a^k, & X & \to & aX \mid bX \mid \lambda. \\ B & \to & Bb \mid \lambda, & & & \end{array}$$

It can be shown that $G_C^{(ck)}$ *is a PSFL(*$k+2$*)-grammar.*

We can also see that the word $¢a^k\$$ *is a word of length* $k+2$ *of* $L\left(G_A^{(ck)}\right)$*, and that it cannot be reduced by any reduction according to* $G_C^{(ck)}$*. This means that* $G_C^{(ck)}$ *is not a PSFL(*$k+1$*)-grammar.*

The next corollary is a consequence of the previous examples.

Corollary 6. *Let* k *be a natural number greater than or equal to* 3*. Then there is a PSFL(*k*)-grammar, which is not a PSFL(*$k-1$*)-grammar.*

3.4. Non-regularity by complete CF-grammars

Example 5. *This example shows that there is a complete CF(¢,$)-grammar that generates a non-regular accepting language. Consider a complete CF(¢,$)-grammar* $G_C^{(5)} = \left(G_A^{(5)}, G_R^{(5)}\right)$ *generating the accepting language* $\{¢a^nb^m\$ \mid n \geq m > 0\}$ *using the following rules:*

$$\begin{array}{lcllcl} S & \to & S_A \mid S_R, & S_R & \to & ¢XbaX \mid ¢A_1\$ \mid ¢C\$, \\ S_A & \to & ¢A\$ \mid ¢B\$, & A_1 & \to & aB_1 \mid B_1 \mid b, \\ A & \to & aB \mid ab, & B_1 & \to & A_1b, \\ B & \to & Ab \mid A, & C & \to & aC \mid \lambda, \\ & & & X & \to & aX \mid bX \mid \lambda. \end{array}$$

We show that $L\left(G_A^{(5)}\right)$ *is a non-regular language by contradiction. Suppose that* $L\left(G_A^{(5)}\right)$ *is a regular language. Then, by a pumping lemma for regular languages, we have a positive constant* k *such that the suffix* $b^k\$$ *of the word* $¢a^kb^k\$ \in L\left(G_A^{(4)}\right)$ *contains a subword which can be pumped, and all the pumped*

words remain in $L\left(G_A^{(5)}\right)$. *However,* \$ *cannot be pumped because all words in* $L\left(G_A^{(5)}\right)$ *contain exactly one occurrence of* \$. *Finally, no occurrence of* b *can be pumped, as the condition* $\geq$ *of the number of* a*'s and the number of* b*'s will be disturbed. That is a contradiction.*

If G_C is a complete CF(¢,\$)-grammar, then both $L(G_A)$ and $L(G_R)$ are context-free languages. How can we see whether those languages are non-regular? See the paper [1] that (partially) answers that question.

4. PSFG-grammars and One-side Grammars

PSFG-grammars introduced in this section are inspired by the notions in [2].

Definition 9. *Let* $G = (N, \Sigma \cup \{¢, \$\}, S, P)$ *be a CF(¢,\$)-grammar,* $x, u_1, v, u_2, y \in \Sigma^*$, $u_1u_2 \neq \lambda$, $A \in N$, *and*

$$S \Rightarrow^* ¢xAy\$ \Rightarrow^+ ¢xu_1Au_2y\$ \Rightarrow^* ¢xu_1vu_2y\$. \tag{1}$$

We say that A *is a* reducing nonterminal, $(¢x, u_1, v, A, u_2, y\$)$ *is a* reducing G-infix, $(¢x, u_1, v, u_2, y\$)$ *is a* pure reducing G-infix, $¢xu_1vu_2y\$ \equiv\!>_G ¢xvy\$$ *is a* G-reduction.

A (pure) reducing G*-infix or a* G*-reduction is* two-side *if both* u_1 *and* u_2 *are nonempty. They are* right-side *(*left-side*, respectively) if* u_1 *(*u_2*, respectively) is empty.*

A (pure) reducing G*-infix or a* G*-reduction is* one-side *if* u_1 *or* u_2 *is empty.*

The relation $\equiv\!>_G^*$ *is the reflexive and transitive closure of the reduction relation* $\equiv\!>_G$.

Note that if $(¢x, u_1, v, A, u_2, y\$)$ is a *reducing* G*-infix*, then for each nonnegative integer i, we have a derivation $S \Rightarrow^* ¢xu_1^ivu_2^iy\$$. This fact was used to define pumping reductions in [2].

One-side and Two-side Reducing CF(¢,\$)-grammars

Let G be a CF(¢,\$)-grammar. We say that G is a *left-side reducing CF(¢,\$)-grammar* if all its reducing infixes are left-side reducing infixes (Definition 9). Similarly, we say that G is a *right-side reducing CF(¢,\$)-grammar* if all its reducing infixes are right-side reducing infixes. We say that G is a *one-side reducing CF(¢, \$)-grammar* if it is either a left-side or right-side reducing CF(¢, \$)-grammar.

Definition 10. *Let* $G_C = (G_A, G_R)$ *be a complete CF(¢, \$)-grammar and* $d > 0$ *be a constant such that for all words* x, x_1, s *all of the following three conditions are satisfied:*

(P) For each $xs \in L(G_C)$ *such that* $|x| > d$, *there is* x_1, $|x| - |x_1| \leq d$ *such that* $xs \equiv\!>_{G_C} x_1s$.
(S) For each $xs \in L(G_C)$ *such that* $|s| > d$, *there is* s_1, $|s| - |s_1| \leq d$ *such that* $xs \equiv\!>_{G_C} xs_1$.
(C) All the G_C*-reductions mentioned in (P) and (S) are right-side or left-side* G_C*-reductions.*

We say that G_C *is a PSFG1(d)-grammar.*

Note that the definition of PSFG1(d)-grammar uses reductions based on the (structural) properties of G_C. A PSFG(d)-grammar need not be one-side complete CF(¢,\$)-grammar, as the conditions of Definition 10 can be satisfied also when G_C enables both left-side and right-side G_C-reductions, or even two-side reductions (see Example 6 below).

Definition 11 ((P, G_C)**-reductions,** (S, G_C)**-reductions).** *Let* $G_C = (G_A, G_R)$ *be a PSFG1(d)-grammar, for some integer* $d > 0$, *and* $x, x_1, u_1, u_2, v, y_1, s$ *be words.*

(gp) Let $xs \in L(G_C)$, $|x| > d$, $x = x_1u_1vu_2y_1$, *where* $In = (x_1, u_1, v, u_2, y_1s)$ *is a pure one-side reducing* G_C*-infix.*
We write $x_1u_1vu_2y_1s \equiv\!>_{P,G_C} x_1vy_1s$ *and say that* $x_1u_1vu_2y_1s \equiv\!>_{P,G_C} x_1vy_1s$ *is a* (P, G_C)-reduction.

(gs) *Let* $xs \in L(G_C)$, $|s| > d$, $s = s_1u_1vu_2y_1$, *where* $In = (xs_1, u_1, v, u_2, y_1)$ *is a pure one-side pumping* G_C*-infix.*
We write $xs_1u_1vu_2y_1 \equiv\!>_{S,G_C} xs_1vy_1$ *and say that* $xs_1u_1vu_2y_1 \equiv\!>_{S,G_C} xs_1vy_1$ *is an* (S, G_C)-reduction.

The relations $\equiv\!>^*_{S,G_C}$ *and* $\equiv\!>^*_{P,G_C}$ *are the reflexive and transitive closures of the relations* $\equiv\!>_{S,G_C}$ *and* $\equiv\!>_{P,G_C}$, *respectively.*

Definition 12 (PSFG(d)-grammars). *Let* $G_C = (G_A, G_R)$ *be a PSFG1(d)-grammar, for some* $d > 0$. *We say that* G_C *is a PSFG(d)-grammar if the following condition (Cg) is fulfilled for all words* x, y, s, s'*:*

(Cg) *If* $xs' \equiv\!>^*_{P,G_C} x's'$, $ys' \equiv\!>^*_{P,G_C} x's'$ *and* $xs \equiv\!>^*_{S,G_C} xs'$, *then* $ys \equiv\!>^*_{S,G_C} ys'$.

Obviously, Condition (Cg) in the definition of PSFG(d)-grammar (Definition 12) corresponds to Condition (A) of the definition of PSFL(d)-grammar (Definition 5).

Lemma 4. *Let* $G_C = (G_A, G_R)$ *be a PSFG(d)-grammar, for some* $d > 0$. *Then* G_C *is a PSFL(d)-grammar.*

Proof: We can see from the definitions that each (S, G_C)-reduction ($\equiv\!>^*_{S,G_C}$) according to PSFG(d)-grammar G_C is an S-reduction ($\rightsquigarrow_{S,G_C}$) according to a PSFL(d)-grammar G_C, and that each (P, G_C)-reduction ($\equiv\!>^*_{P,G_C}$) according to PSFG(d)-grammar G_C is a P-reduction ($\rightsquigarrow_{P,G_C}$) according to PSFL(d)-grammar G_C. Therefore, each PSFG(d)-grammar satisfying conditions (P), (S), and (C) of Definition 10 satisfies the conditions (Ap), (As), (Rp), and (Rs) for PSFL(d)-grammar (Definition 3). Therefore, each PSFG1(d)-grammar is a PSFL1(d)-grammar. Further, Condition (Cg) of Definition 12 implies Condition (A) of Definition 5. Thus, each PSFG(d)-grammar is a PSFL(d)-grammar. □

Using Theorem 1, we obtain the following corollary.

Corollary 7. *Let* $G_C = (G_A, G_R)$ *be a PSFG(d)-grammar. Then the languages* $L(G_A)$ *and* $L(G_R)$ *are regular.*

Theorem 2. *A language* L *is a regular language if and only if there exists a PSFG(d)-grammar* $G_C = (G_A, G_R)$ *such that* $L = L_{\mathrm{in}}(G_A)$.

Proof: According to Corollary 7, for each PSFG(d)-grammar, the languages $L(G_A)$ and $L(G_R)$ are regular. It remains to show that if L is a regular language, then there exists a PSFG(d)-grammar $G_C = (G_A, G_R)$ such that $L = L_{\mathrm{in}}(G_A)$. Then, there exists a deterministic (complete) finite state automaton $M = (Q, \Sigma, \delta, q_0, F)$ such that $L = L(M)$. Obviously, if we replace the set of final states F of the automaton M with $Q \setminus F$, we get a finite state automaton $(Q, \Sigma, \delta, q_0, Q \setminus F)$ that accepts the language $\Sigma^* \setminus L$.

Based on the finite automaton M, we can construct a complete CF(¢, \$)-grammar $G_C = (N, \Sigma \cup \{¢, \$\}, S, P)$, where $N = \{S, S_A, S_R\} \cup Q \cup Q'$, where $Q' = \{q' \mid q \in Q\}$ is a set of marked versions of the states from Q, the symbols S, S_A, and S_R are new nonterminals, $\{S, S_A, S_R\} \cap (Q \cup Q') = \emptyset$, and P contains the following rules:

$$
\begin{array}{ll}
S \to S_A \mid S_R, & \\
S_A \to ¢q_0\$, & \\
S_R \to ¢q'_0\$, & \\
X \to xY & \text{if } X, Y \in Q \text{ and } \delta(X, x) = Y, \\
X' \to xY' & \text{if } X, Y \in Q \text{ and } \delta(X, x) = Y, \\
X \to \lambda, & \text{if } X \in F, \\
X' \to \lambda, & \text{if } X \in Q \setminus F.
\end{array}
$$

Each derivation of a word $¢w_1w_2\cdots w_n\$ \in L(G_C)$, where $w_1,\ldots,w_n \in \Sigma$, for some $n \geq 0$, according to the grammar G_C is one of the following two forms:

$$\begin{aligned} S \Rightarrow_{G_C} S_A \Rightarrow_{G_C} ¢p_0\$ \Rightarrow_{G_C} ¢w_1p_1\$ \Rightarrow_{G_C} ¢w_1w_2p_2\$ \Rightarrow_{G_C} \cdots \\ \Rightarrow_{G_C} ¢w_1w_2\cdots w_np_n\$ \Rightarrow_{G_C} ¢w_1\cdots w_n\$, \qquad (2) \\ S \Rightarrow_{G_C} S_R \Rightarrow_{G_C} ¢p'_0\$ \Rightarrow_{G_C} ¢w_1p'_1\$ \Rightarrow_{G_C} ¢w_1w_2p'_2\$ \Rightarrow_{G_C} \cdots \\ \Rightarrow_{G_C} ¢w_1w_2\cdots w_np'_n\$ \Rightarrow_{G_C} ¢w_1\cdots w_n\$. \qquad (3) \end{aligned}$$

where $p_0,\ldots,p_n \in Q$, $p_0 = q_0$, $p_n \in F$, $p'_0,\ldots,p'_n \in Q'$, $p'_0 = q'_0$, $p'_n \in F' = \{q' \mid q \in Q_L \setminus F_L\}$. Then, all reducing G_C-infixes are of the following form:

$$(¢x, u_1, v, A, u_2, y\$) = (¢x, u_1, v, A, \lambda, \$),$$

where x, u_1, v, u_2, y are words from Σ^*, $¢xu_1vu_2y\$ \in L(G_C)$, $u_2 = \lambda$, $y = \$$, and either $A \in Q$ or $A \in Q'$. Evidently, G_C is a left-side reducing complete CF($¢, \$$)-grammar. Moreover, we will show that G_C is a PSFG(d)-grammar, where $d = |Q|$.

At first, let us check that G_C is a PSFG1(d)-grammar. For some words x and s, if $xs \in L(G_C)$ and $|x| > d$, the beginning of the derivation of the word $w = ¢w_1\cdots w_n\$ = xs$ is in one of the following two forms:

$$\begin{aligned} S \Rightarrow S_A \Rightarrow ¢p_0\$ \Rightarrow ¢w_1p_1\$ \Rightarrow ¢w_1w_2p_2\$ \Rightarrow \cdots \Rightarrow ¢w_1w_2\cdots w_np_d\$, \\ S \Rightarrow S_R \Rightarrow ¢p'_0\$ \Rightarrow ¢w_1p'_1\$ \Rightarrow ¢w_1w_2p'_2\$ \Rightarrow \cdots \Rightarrow ¢w_1w_2\cdots w_dp'_d\$, \end{aligned}$$

where $p_0,\ldots,p_d \in Q$, $p_0 = q_0$, $p'_0,\ldots,p'_d \in Q'$, $p'_0 = q'_0$, and $¢w_1\cdots w_d$ is a prefix of x. Since $|Q| = d$, there exist $i, j \in \{0,\ldots,d\}$, $i < j$, for which $p_i = p_j$ or $p'_i = p'_j$. Then,

$$(¢w_1\cdots w_i, w_{i+1}\cdots w_j, w_{j+1}\cdots w_n, A, \lambda, \$)$$

is a reducing G_C-infix, where either $A = p_i = p_j \in Q$ or $A = p'_i = p'_j \in Q'$. In both cases, $¢w_1\cdots w_n\$ = ¢w_1\cdots w_iw_{i+1}\cdots w_jw_{j+1}\cdots w_n\$ \equiv_{>P,G_C} ¢w_1\cdots w_iw_{j+1}\cdots w_n\$$. Hence, there exists a word x' obtained by omitting $w_{i+1}\cdots w_j$ from x such that $xs \equiv_{>P,G_C} x_1s$ and $|x| - |x_1| \leq d$, and condition (P) of Definition 10 is satisfied.

Similarly, we can find a repeating state in the last d steps of the derivation of the form (2) and (3) and show that condition (S) of Definition 10 is true.

Condition (C) of Definition 10 follows from the fact that all reducing G_C-infixes are left-side.

Altogether, we see that G_C is a PSFG1(d)-grammar.

Finally, let us show that the condition (Cg) of the definition of PSFG(d)-grammar (Definition 12) holds.

Let $\alpha\beta \equiv_{>G_C} \alpha'\beta$ be a G_C-reduction. Then, there exists a reducing G_C-infix $(¢\alpha_1, u_1, v_1\beta_1, A, \lambda, \$)$ for some words $\alpha_1, u_1, v_1, \beta_1 \in \Sigma^*$ and a nonterminal A such that $\alpha = ¢\alpha_1u_1v_1$, $\beta = \beta_1\$$, $\alpha' = ¢\alpha_1v_1$, and we have the following derivation

$$S \Rightarrow^*_{G_C} ¢\alpha_1A\$ \Rightarrow^*_{G_C} ¢\alpha_1u_1A\$ \Rightarrow^*_{G_C} ¢\alpha_1u_1v_1B\$ = \alpha B\$ \Rightarrow^*_{G_C} ¢\alpha_1u_1v_1\beta_1\$ = \alpha\beta,$$

for some nontermainal $B \in Q \cup Q'$. Then the following derivation is possible

$$S \Rightarrow^*_{G_C} ¢\alpha_1A\$ \Rightarrow^*_{G_C} ¢\alpha_1v_1B\$ = \alpha' B\$ \Rightarrow^*_{G_C} ¢\alpha_1v_1\beta_1\$ = \alpha'\beta.$$

This means that β_1 is generated from the nonterminal B in both of the above derivations and

$$\begin{aligned} S \Rightarrow^*_{G_C} \alpha B\$ \Rightarrow^*_{G_C} \alpha\beta, \\ S \Rightarrow^*_{G_C} \alpha' B\$ \Rightarrow^*_{G_C} \alpha'\beta. \end{aligned}$$

Using induction on the number of steps in the sequence of reductions $\alpha\beta \equiv^*_{>G_C} \alpha'\beta$, we can show that there exists a nonterminal B and derivations

$$\begin{aligned} S \Rightarrow^*_{G_C} \alpha B\$ \Rightarrow^*_{G_C} \alpha\beta \text{ and} \\ S \Rightarrow^*_{G_C} \alpha' B\$ \Rightarrow^*_{G_C} \alpha'\beta. \end{aligned}$$

Similarly, if $\alpha\beta \equiv>^*_{G_C} \alpha\beta'$, there exists a nonterminal C such that

$$S \Rightarrow^*_{G_C} \alpha C\$ \Rightarrow^*_{G_C} \alpha\beta \text{ and}$$
$$S \Rightarrow^*_{G_C} \alpha C\$ \Rightarrow^*_{G_C} \alpha\beta'.$$

Let $xs' \equiv>^*_{P,G_C} x's'$, $ys' \equiv>^*_{P,G_C} x's'$ and $xs \equiv>^*_{S,G_C} xs'$, for some words x, y, s, s'. Using the above observations, we know that there exist nonterminals B, C, D such that

$$S \Rightarrow^*_{G_C} xB\$ \Rightarrow^*_{G_C} xs', \quad S \Rightarrow^*_{G_C} yC\$ \Rightarrow^*_{G_C} ys', \quad S \Rightarrow^*_{G_C} xD\$ \Rightarrow^*_{G_C} xs,$$
$$S \Rightarrow^*_{G_C} x'B\$ \Rightarrow^*_{G_C} x's', \quad S \Rightarrow^*_{G_C} x'C\$ \Rightarrow^*_{G_C} x's', \quad S \Rightarrow^*_{G_C} xD\$ \Rightarrow^*_{G_C} xs'.$$

However, G_C is unambiguous. Therefore, from $S \Rightarrow^*_{G_C} x'B\$$ and $S \Rightarrow^*_{G_C} x'C\$$ it follows that $B = C$. From $S \Rightarrow^*_{G_C} xB\$$ and $S \Rightarrow^*_{G_C} xD\$$ it follows that $B = D$. Then

$$S \Rightarrow^*_{G_C} yB\$ \Rightarrow^*_{G_C} ys \text{ and}$$
$$S \Rightarrow^*_{G_C} yB\$ \Rightarrow^*_{G_C} ys',$$

which implies $ys \equiv>^*_{G_C} ys'$, and the condition (Cg) is satisfied. Hence, G_C is a PSFG(d)-grammar.

This completes the proof that G_C is a PSFG(d)-grammar. □

4.1. Separation Results

Example 6. *Let $G_C^{(1)} = (\{S, S_A, S_R, A, B\}, \{a, b, ¢, \$\}, S, R)$ be a complete CF(¢,$)-grammar with the following set of rules:*

$$\begin{array}{lcllcl} S & \to & S_A \mid S_R, & A & \to & aAb \mid ab, \\ S_A & \to & ¢A\$ \mid ¢B\$, & B & \to & aB \mid bB \mid \lambda. \end{array}$$

The grammar $G_C^{(1)}$ is a complete CF(¢,$)-grammar $G_C^{(1)} = \left(G_A^{(1)}, G_R^{(1)}\right)$, where $G_A^{(1)}$ has starting symbol S_A, $G_R^{(1)}$ has starting symbol S_R, and obviously, $L\left(G_C^{(1)}\right) = L\left(G_A^{(1)}\right) = \{¢\} \cdot \{a, b\}^* \cdot \{\$\}$, and $L\left(G_R^{(1)}\right) = \emptyset$.

As $(¢aa, a, ab, A, b, bb\$)$ is a two-side reducing infix by $G_A^{(1)}$, and the word $w = ¢aaaabbbb\$$ is in $L\left(G_A^{(1)}\right)$, the grammar $G_A^{(1)}$ is a two-side reducing grammar. However, the word w can also be derived using rules of $G_A^{(1)}$ that do not contain the nonterminal A, and all corresponding infixes are left-side infixes.

We can see that according to Definition 5 and Definition 12 the complete CF(¢,$)-grammar $G_C^{(1)} = \left(G_A^{(1)}, G_R^{(1)}\right)$ is a PSFL(3)-grammar and a PSFG(3)-grammar that is not a one-side reducing grammar.

Actually, we could omit all rules that include A from the grammar, and we would obtain a grammar that generates the same language as the original grammar and is left-side reducing.

The next corollary is a consequence of Example 6.

Corollary 8. *There is a PSFG(3)-grammar and PSFL(3)-grammar $G_C^{(1)}$ that is not a one-side reducing CF(¢,$)-grammar.*

Example 7. *Let us consider the complete CF(¢,$)-grammar $G_C^{(2)} = \left(G_A^{(2)}, G_R^{(2)}\right)$ generating the accepting language $\{¢a^n b^m\$ \mid n + m \geq 0\}$ using the following rules:*

$$\begin{array}{lcllcl} S & \to & S_A \mid S_R, & B_1 & \to & A_1 b \mid b, \\ S_A & \to & ¢A\$ \mid ¢D\$ \mid ¢B_1\$ \mid ¢\$, & D & \to & aDb \mid ab, \\ A & \to & aB \mid a, & S_R & \to & ¢C\$, \\ B & \to & Ab \mid A \mid \lambda, & C & \to & XCX \mid ba, \\ A_1 & \to & aB_1 \mid B_1 \mid \lambda, & X & \to & aX \mid bX \mid \lambda. \end{array}$$

$G_C^{(2)} = \left(G_A^{(2)}, G_R^{(2)}\right)$ *is a PSFL(4)-grammar according to Definition 12.* $G_A^{(2)}$ *is an unambiguous context-free grammar. We can see that the words generated with the help of* A *by* $G_A^{(2)}$ *are from the set* $\{¢a^nb^m\$ \mid n > m, m \geq 0\}$*, the words generated by* $G_A^{(2)}$ *with the help of* B_1 *are from the set* $\{¢a^nb^m\$ \mid n < m, n \geq 0\}$*, and the words generated by* $G_A^{(2)}$ *with the help of* D *are from the set* $\{¢a^nb^n\$ \mid n > 0\}$*.*

There is no suffix reduction $\equiv>_{S,G_C^{(2)}}$ *within the subword* b^n *of any word of the form* $¢a^nb^n\$$ *where* $n > 0$*. Note that* $¢a^nb^n\$ \in L\left(G_A^{(2)}\right)$ *for* $n \geq 0$*. This means that* $G_C^{(2)}$ *is not a PSFG(*n*)-grammar for any natural number* n*. This observation also emphasizes the difference between the two types of reductions* $\equiv>$ *and* $\rightsquigarrow$ *used in this article for complete CF(¢,$)-grammars.*

The next corollary is a consequence of Example 7.

Corollary 9. *There is a PSFL(4)-grammar* $G_C^{(2)}$ *that is not a PSFG(d)-grammar for any natural number* d*.*

4.2. A Hierarchy of PSFG(d)-grammars

Using some of the above examples, we can show several hierarchy results.

Corollary 10. *There is a PSFG(2)-grammar, and there is no PSFG(1)-grammar.*

Proof: The grammar $G_C^{(c2)}$ from Example 1 is a PSFG(2)-grammar. Obviously, there is no PSFG(1)-grammar. □

Corollary 11. *There is a PSFG(3)-grammar, which is not a PSFG(2)-grammar.*

Proof: The grammar $G_C^{(c3)}$ from Example 2 is a PSFG(3)-grammar, and it is not a PSFG(2)-grammar. □

Corollary 12. *There is a PSFG(4)-grammar, which is not a PSFG(3)-grammar.*

Proof: The grammar $G_C^{(c4)}$ from Example 3 is a PSFG(4)-grammar, and it is not a PSFG(3)-grammar. □

Corollary 13. *Let* k *be a natural number greater than* 0*. Then there is a PSFG(k+2)-grammar, which is not a PSFG(k+1)-grammar.*

Proof: The grammar $G_C^{(ck)}$ from Example 3 is a PSFG(k+2)-grammar, and it is not a PSFG(k+1)-grammar. □

5. Conclusions and Future Work

We believe that the next conjecture can be proved as a non-trivial consequence of Theorem 1 and the construction of a complete CF(¢,$)-grammar in the proof of Theorem 2.

Conjecture 1. *Let* $G_C = (G_A, G_R)$ *be a complete CF(*$¢, \$$*)-grammar, and* $L(G_A)$ *be a regular language. Then* G_C *is a PSFL(*d*)-grammar for some natural number* d*, where* d *is determined by* G_C*.*

While PSFL(c)-grammars depend on the properties of the generated language, but not on the structure of the grammar, PSFG(d)-grammars are based directly on the properties of the grammars. We have shown that the notion of PSFL(c)-grammar is more general than PSFG(c)-grammar. Nevertheless, we have shown a hierarchy based on the parameter c both for PSFL(c)- and PSFG(c)-grammars.

PSFL(c)-grammars and PSFG(d)-grammars can be seen as special types of Marcus contextual grammars; see [12].

We believe that any relaxation of the conditions for PSFL(c)-grammars leads to non-regularity of the considered grammars.

Until now, we have not studied the computability and decidability questions related to complete CF(¢, \$)-grammars. In the future, we will focus on these questions, and one of our plans is to study restarting automata (see [9]) controlled by complete CF(¢, \$)-grammars.

Acknowledgement

Marta Vomlelová thanks for the support from the project 25-18003S of the Czech Science Foundation.

Declaration on Generative AI

The authors have not employed any Generative AI tools.

References

[1] M. Plátek, F. Mráz, D. Pardubská, M. Procházka, Non-regularity of complete CF(¢, \$)-grammars, in: L. Ciencialová, M. Holeňa, R. Jajcay, T. Jajcayová, F. Mráz, D. Pardubská, M. Plátek (Eds.), Proceedings of the 24th Conference Information Technologies – Applications and Theory (ITAT 2024), volume 3792 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2024, pp. 75–80. URL: https://ceur-ws.org/Vol-3792/paper8.pdf.

[2] F. Mráz, D. Pardubská, M. Plátek, M. Procházka, M. Vomlelová, Complete CF(¢, \$)-grammars, pumping reductions, and regularity., in: B. Brejová, L. Ciencialová, M. Holeňa, R. Jajcay, T. Jajcayová, M. Lexa, F. Mráz, D. Pardubská, M. Plátek (Eds.), Proceedings of the 25th Conference Information Technologies – Applications and Theory (ITAT 2025), CEUR Workshop Proceedings, CEUR-WS.org, 2025, pp. 205–214. URL: https://ceur-ws.org/Vol-4092/paper16.pdf.

[3] P. Jančar, J. Šíma, The simplest non-regular deterministic context-free language, in: F. Bonchi, S. J. Puglisi (Eds.), 46th International Symposium on Mathematical Foundations of Computer Science, MFCS 2021, August 23–27, 2021, Tallinn, Estonia, volume 202 of *LIPIcs*, Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021, pp. 63:1–63:18. doi:10.4230/LIPIcs.MFCS.2021.63.

[4] J. Šíma, M. Plátek, One analog neuron cannot recognize deterministic context-free languages, in: T. Gedeon, K. W. Wong, M. Lee (Eds.), Neural Information Processing – 26th International Conference, ICONIP 2019, Sydney, NSW, Australia, December 12–15, 2019, Proceedings, Part III, volume 11955 of *Lecture Notes in Computer Science*, Springer, 2019, pp. 77–89. doi:10.1007/978-3-030-36718-3_7.

[5] M. Lopatková, M. Plátek, P. Sgall, Towards a formal model for functional generative description: Analysis by reduction and restarting automata, Prague Bull. Math. Linguistics 87 (2007) 7–26. URL: http://ufal.mff.cuni.cz/pbml/87/lopatkova-et-al.pdf.

[6] P. Jančar, F. Mráz, M. Plátek, J. Vogel, On monotonic automata with a restart operation, J. Autom. Lang. Comb. 4 (1999) 287–311. doi:10.25596/jalc-1999-287.

[7] F. Otto, Restarting automata, in: Z. Ésik, C. Martín-Vide, V. Mitrana (Eds.), Recent Advances in Formal Languages and Applications, volume 25 of *Studies in Computational Intelligence*, Springer, 2006, pp. 269–303. URL: https://doi.org/10.1007/978-3-540-33461-3_11.

[8] M. A. Harrison, Introduction to Formal Language Theory, Addison-Wesley, USA, 1978.

[9] F. Otto, Restarting Automata: The Standard Type of Restarting Automaton and Its Variants, Springer Nature Switzerland, Cham, 2024. URL: https://doi.org/10.1007/978-3-031-70094-1. doi:10.1007/978-3-031-70094-1.

[10] J. Hopcroft, J. Ullman, Introduction to Automata Theory, Languages, and Computation, Addison-Wesley, N. Reading, MA, 1980.

[11] J. E. Hopcroft, J. D. Ullman, Formal languages and their relation to automata, Addison-Wesley series in computer science and information processing, Addison-Wesley, 1969. URL: https://www.worldcat.org/oclc/00005012.

[12] G. Paun, Marcus contextual grammars, volume 67 of *Studies in Linguistics and Philosophy*, Kluwer, Dordrecht, Boston, London, 1997.