VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

STROMOVĚ OMEZENÉ GRAMATIKY JAKO PROSTŘEDEK PRO DŮKAZ BEZKONTEXTOVOSTI

CONTEXT-FREENESS RESULTING FROM TREE-RESTRICTED GRAMMARS

DISERTAČNÍ PRÁCE
PHD THESIS

AUTOR PRÁCE — Ing. ONDŘEJ SOUKUP
AUTHOR

VEDOUCÍ PRÁCE — prof. RNDr. ALEXANDER MEDUNA, CSc.
SUPERVISOR

BRNO 2017

Abstrakt

Tato disertační práce představuje derivační stromy několika různých typů gramatik ve zobecněné Kurodově normální formě; jmenovitě obecné a regulárně řízené gramatiky, gramatiky s rozptýleným kontextem a spolupracující distribuované gramatické systémy. Definuje jednoduché stromové rysy založené na kontextových vlastnostech jednotlivých diskutovaných gramatik a dokazuje, že pokud existuje limitující konstanta k taková, že každá věta generovaného jazyka L odpovídá řetězci listových uzlů derivačního stromu, ve kterém je výskyt definovaných stromových rysů omezen konstantou k, jazyk L je ve skutečnosti bezkontextový. Tato práce dále ukazuje, že dosažený výsledek představuje silný nástroj důkazu bezkontextovosti jazyka. Vše je doplněno příklady praktického využití nástroje.

Abstract

The present thesis introduces derivation trees for several different types of grammars in generalized Kuroda normal forms; namely, general, regular-controlled, and scattered context grammars and cooperating distributed grammar systems. It defines simple tree-based properties related to non-context-free properties of all grammars in question and shows that if there exists a limiting constant k such that every sentence in the generated language L is the frontier of a derivation tree in which the number of occurrences of the defined tree-based properties is limited by k, the language L is in fact context-free. The thesis explains that this result represents a powerful tool for showing languages to be context-free. It also provides illustrations and examples which sketch how to apply this tool in practice.

Klíčová slova

Obecné gramatiky, regulárně řízené bezkontextové gramatiky, gramatiky s rozptýleným kontextem, spolupracující distribuované gramatické systémy, normalní formy, derivační stromy, omezené derivační stromy, bezkontextovost, bezkontextovost konečného indexu.

Keywords

General grammars, regular-controlled context-free grammars, scattered context grammars, cooperating distributed grammar systems, normal forms, derivation trees, restricted derivation trees, context-freeness, context-freeness of finite index.

Citace

Ondřej Soukup: Context-Freeness Resulting from Tree-Restricted Grammars, disertační práce, Brno, FIT VUT v Brně, 2017

Context-Freeness Resulting from Tree-Restricted Grammars

Declaration

I declare that I created this thesis individually under the supervision of prof. Alexander Meduna and partially under supervision of prof. Erzsébet Csuhaj-Varjú. I mentioned all the literature and publications, from which I drew.

. . . .
Ondřej Soukup
May 29, 2017

Acknowledgment

First of all, I would like to thank prof. Alexander Meduna for his professional guidance and supervision of this work, his continuous initiative, and enthusiasm thanks to which this project could be created. Part of this thesis is also based on the work written in cooperation with prof. Erzsébet Csuhaj-Varjú to whom I am very thankful too. Considerable thanks also belong to Zbyněk Křivka for his significant help and many pieces of advice. Last but not least am extremely grateful to my family for long years of support and care.

This work was partially supported by Erasmus+ grant.

Contents

1 Introduction **2**

2 Languages and Their Models **8**
- 2.1 Languages 8
- 2.2 Grammars 10
- 2.3 Automata 19

3 Graph-Based Representation of Derivation **20**
- 3.1 Graphs and Trees 20
- 3.2 Derivation Trees 22

4 Proof Techniques in Formal Language Theory **32**
- 4.1 Positive Proofs 32
- 4.2 Negative Proofs 37

5 Hierarchy of Language Families **47**

6 Tree-Restricted Grammars **51**
- 6.1 General Grammars 51
- 6.2 Regular-Controlled Grammars 54
- 6.3 Scattered Context Grammars 64
- 6.4 Cooperating Distributed Grammar Systems 75

7 How to Prove Context-Freeness **85**

8 Conclusion **92**

A Publications **99**

Chapter 1

Introduction

Formal language theory has always intensively struggled to establish conditions under which certain grammars generate only a proper subfamily of the family of languages generated by this type of grammars as a whole because results like this often significantly simplify proofs that some languages are members of the subfamily. To illustrate by a specific example, consider the well-known workspace theorem for context-sensitive languages of general grammars (see Theorem 4.1.1), which fulfills a crucially important role in the grammatically oriented theory of formal languages as a whole (see Theorem III.10.1 in [50]). This theorem represents a powerful tool to demonstrate that if a general grammar G generates each of its sentences by a derivation satisfying a prescribed condition (specifically, this condition requires that there is a positive integer k such that G generates every sentence y in the generated language $\mathrm{L}(G)$ by a derivation in which every sentential form x satisfies $|x| \leq k|y|$), then $\mathrm{L}(G)$ is a member of the context-sensitive language family. Regarding the membership in the context-free language family, however, formal language theory lacks a result like this. To fill this gap, the present thesis establishes several tree-based conditions so every grammar in question satisfying these conditions generates a member of the context-free language family.

More specifically, we discuss general, regular-controlled context-free, and scattered context grammars and cooperating distributed grammar systems. In general, all of these types of grammars (for definitions see Section 2.2) are significantly stronger than ordinary context-free grammars. In fact, general and scattered context grammars are as strong as Turing machines and, therefore, they are computationally complete. Regular-controlled grammars are as powerful as matrix grammars—that is, they generate the family of matrix languages. The generative power of cooperating distributed grammar systems ranges from the power of context-free grammars up to matrix grammars or extended tabled zero-sided Lindermayer systems depending on the derivation mode respectively (see [28, 35, 42, 48, 50] for details). Surprisingly, the present thesis demonstrates that under some very natural and simple conditions placed upon their derivation trees, their power significantly lessens. As a matter of fact, under these conditions, they generate only the family of context-free languages—or even context-free languages of finite index.

In theory of formal languages we have typically a numerous different ways how to disprove that some language belongs to a certain family of languages and, thus, obtain a negative proof of a membership in this family of languages (for details see Section 4.2). The most significant examples are indisputably various pumping lemmas; namely, for regular, linear, and context-free languages. They introduce necessary (rarely also sufficient) conditions for a language to belong to the language family in question such that for a language L and a sufficiently long sentence $w \in L$, there are certain substrings of w which may be so called pumped to again obtain a sentence in L. However, we usually use these lemmas counterwise showing that for a language L and any sufficiently long sentence w, there do not exist substrings specified by the given lemma which can be pumped to obtaining sentences in L. To give another example, there are several proof techniques to show that some languages are not recursively enumerable and, so, beyond the power of general grammars; specifically, reduction or diagonalization. On the other hand, there are only a few proof techniques to obtain positive proof of membership of a language in a language family. The most significant example is definitely the workspace theorem explained before. Especially, except for example constructing a context-free grammar, there is no general proof technique for proving context-freeness. Though, this thesis aims to contribute to the subject of proving context-freeness by showing that some non-context-free grammars under some simple derivation-tree-based restrictions generate context-free languages.

The following simple but natural idea stands in the background of this thesis.

Basic Idea. The vast majority of non-context-free grammars is in some way based upon the very basic concept of context-free rules. Additionally, they introduce some mechanisms to increase their generative power. Some of them allow the rules to be applied only in a certain context of neighbouring symbols, restrict the mutual order of the rule applications, perform parallel applications of several context-free rules at once, or alternate between several sets of context-free rules under the given conditions. Each of these additional mechanisms provides the power beyond context-free grammars, since it may be applied at any time during the whole derivation process. It can be also very naturally captured by the specific properties of corresponding derivation trees. Then, if we restrict these derivation trees, so the number of these specific properties occurring within them is less or equal to some constant value, the generative power probably significantly decreases.

The initial thoughts indicated that the resulting generated family of languages for any of these restricted grammars is precisely the family of context-free languages. Later in Chapter 6, we provide rigorous proofs and detailed explanation that this basic idea in fact holds true; except for regular-controlled context-free grammars in which case the restriction is even more significant. To give an insight we sketch the results achieved for all grammars in question.

I. First, we introduce tree-based conditions for context-freeness of general grammars. Recall that a general grammar G is in Kuroda normal form (see Section 2.2) if any rule satisfies one of these forms

$$AB \to CD, \quad A \to BC, \quad A \to B, \quad A \to a, \quad A \to \varepsilon$$

where A, B, C, D are nonterminals, a is a terminal, and ε is the empty string. We define the notion of a derivation tree t graphically representing a derivation in G by analogy with this notion in terms of an ordinary context-free grammar (for details see Section 3.2). In addition, however, we introduce context-dependent pairs of nodes in t as follows. In t, two paths are neighbouring if no other path occurs between them. Let p and q be two neighbouring paths in t. Let p contain a node k with a single child l, where k and l are labelled with A and C, respectively, and let q contain a node m with a single child n, where m and n are labelled with B and D, respectively. Let this four-node portion of t; consisting of k, l, m, and n; graphically represent an

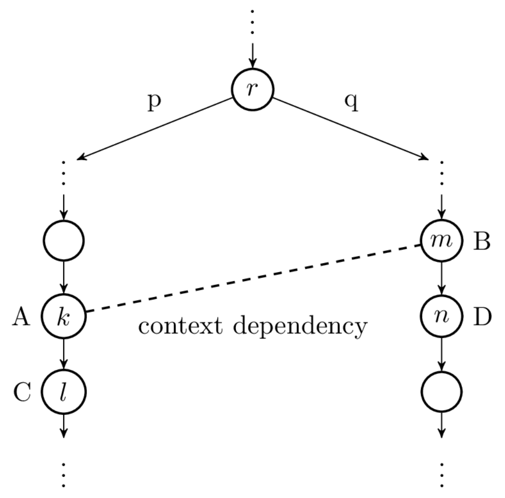


Figure 1.0.1: Illustration of context dependency in derivation tree of general grammar.

application of $AB \to CD$. Then, k and m are a context-dependent pair of nodes (see Fig. 1.0.1).

The present thesis proves that the language of G, $\mathrm{L}(G)$, is context-free if there is a constant k such that every $w \in \mathrm{L}(G)$ is the frontier of a derivation tree d in which any pair of neighbouring paths contains k or fewer context-dependent pairs of nodes.

II. Second, we present tree-based conditions for regular-controlled grammars to generate context-free languages. Consider a context-free grammar G with the following rules

$$S \to AB,\ A \to C,\ B \to D$$

and the next derivation

$$S \Rightarrow AB \Rightarrow CB \Rightarrow CD.$$

Let us look at the process of the derivation from the perspective of the derivation tree. During the first step a branching node is introduced. Then, the derivation continues into the left branch. However, the last derivation step takes place within the different branch; which is the right one. We call this phenomenon path-change and discuss it in terms of regular-controlled grammars.

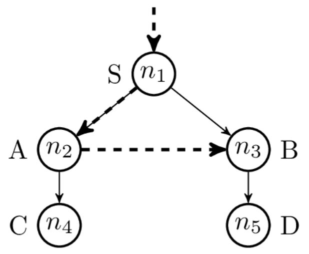


Figure 1.0.2: Illustration of path-change in derivation tree. The dashed lines denote the path of the derivation. Notice that the consecutive rewritings of the node labelled A and the node labelled B take place within the different branches of the derivation tree.

In essence, we put restrictions on the number of path-changing derivation steps. It is proved that the language generated by a regular-controlled grammar is context-free—in fact context-free of index k—if there is a constant k such that every sentence w in the generated language is the frontier of a derivation tree corresponding to some derivation within which there are k or fewer path-changing derivation steps. Of course, this k is an upper bound and the minimal index is possibly lower.

Since the tree-based restrictions are independent of the control mechanism, the achieved result holds for well-known matrix grammars (see [1]) as well.

III. Third, we turn our attention to the scattered context grammars. Recall that a scattered context grammar G is in binary form (see Section 2.2) if in G any rule satisfies one of these forms

$$(A, B) \to (C, D),\ (A) \to (BC),\ (A) \to (X)$$

where A, B, C, D are nonterminals, X is a nonterminal, terminal, or the empty string. We define the notion of a derivation tree t graphically representing a derivation in G by analogy with this notion in terms of an ordinary context-free grammar. In addition, however, we introduce context-dependent pairs of nodes in t as follows. Assume that by applying $(A, B) \to (C, D)$, G simultaneously rewrites A and B to C and D, respectively. This application is graphically depicted in Fig. 1.0.3, where nodes k, m, l, and n are labelled with A, B, C, and D, respectively. Then, k and m is a context-dependent pair of nodes.

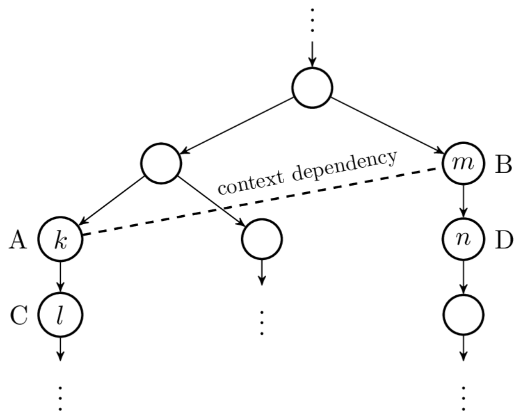


Figure 1.0.3: Illustration of context dependency in derivation tree of scattered context grammar.

Based upon the context-dependent pairs of nodes, the present thesis places a simple restriction upon derivation trees for a scattered context grammar G and demonstrates that under this restriction, its language $\mathrm{L}(G)$ is context-free. In essence, this restriction, sketched in Fig. 1.0.4, requires the existence of a constant $k \geq 0$ such that for any sentence $x \in \mathrm{L}(G)$, there is a derivation tree t with frontier x so t can be divided into a set of connected subgraphs satisfying conditions (i) and (ii):

(i) any pair of context-dependent nodes contained in t occurs within one of these subgraphs, and

(ii) none of these subgraphs contains more than k pairs of context-dependent nodes.

It is worth pointing out that the number of these connected subgraphs and, thus, the total number of context dependencies is not limited at all.

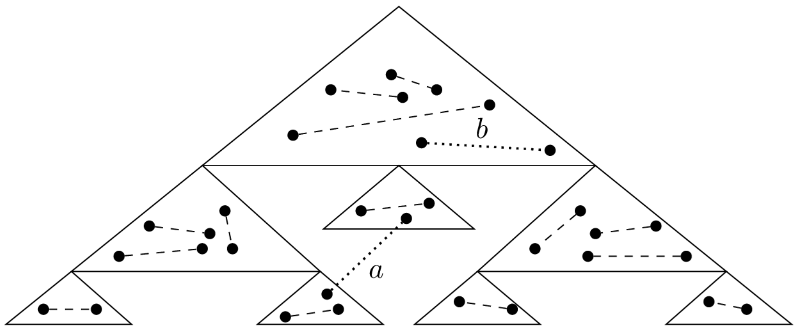


Figure 1.0.4: A sketch of the context-dependence-based restrictions in scattered context grammars. The graph represents derivation tree, where triangles are its subgraphs according to some division. Dashed lines depict the context dependencies. If we add context dependency a, restriction (i) is violated since the affected nodes occur in the different subgraphs of the division of the derivation tree. Suppose that $k = 3$. Then, adding context dependency b violates restriction (ii) since there occurs a subgraph with more than k context dependent pairs of nodes. However, for $k > 3$, adding context dependency b is still legal since both restrictions remain satisfied.

IV. Finally, as the fourth type of grammatical models we cover cooperating distributed grammar systems. They represent a composition of several context-free grammars called components with common alphabets but distinct rule sets. While generating a single sentence, the components alternate under the conditions specified by so called derivation mode. Consider a cooperating distributed grammar system with n components and a derivation

$$u \Rightarrow_i^\psi v \Rightarrow_j^\psi w,$$

where ψ is a derivation mode and $1 \leq i, j \leq n$, $i \neq j$. Then, the first derivation step is performed by component i and the second by different component j. We call this phenomenon component change. In the sense of derivation trees, for a cooperating distributed grammar system G and a sentence w with the derivation tree t there are several layers within t from the root to the leaf nodes, where every layer corresponds to a different component of G (as depicted in Fig. 1.0.5).

These layers are in fact specific cuts of the derivation tree. We denote them component-change cuts. Based on this notion, the thesis shows that restricting the number of possible component-change cuts within the derivation trees of sentences of a cooperating distributed grammar system by a constant value results in generation of context-free language regardless of the used mode of derivations.

Apart from their obvious theoretical value, these results may be of some interest in practice, too. Specifically, some language processors, such as parsers, frequently require that the languages processed by them are context-free. As obvious, the results stated above may fulfill a useful role during the verification of this requirement.

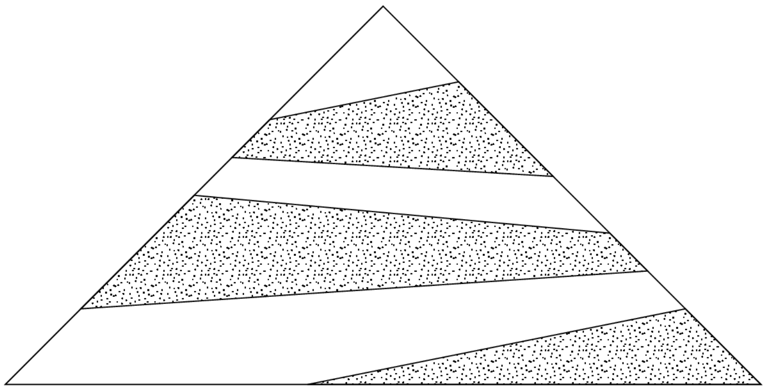


Figure 1.0.5: Illustration of component-change-based layers of derivation trees in cooperating distributed grammar systems. Consider a grammar system with two components and some derivation whose derivation tree is illustrated by the triangle. First component generates sentential form from the start symbol, which corresponds to the topmost layer of the derivation tree. Then, the components change and the second one continues with the generation from the current sentential form. It corresponds to the second topmost grey-dotted layer of the derivation tree, etc.

The thesis is organized as follows. After this introductory chapter, Chapter 2 gives all the necessary terminology concerning languages, grammars, and automata. In the case that the reader is already familiar with theory of formal languages, the chapter may be skipped and serve later as a reference for the notion. Then, in Chapter 3, we first introduce some basic knowledge of graphs and trees to next put it in connection with grammar-based notion of derivation introducing derivation trees which denote hierarchical structure of rewritings in a derivation of a sentence. Chapter 4 presents important proof techniques in formal language theory demonstrating, on one hand, how to prove that a language belongs to a certain family of languages and, on the other hand, how to disprove a membership of a language in a certain language family. It also explains current deficiencies in this area of research. The following Chapter 5 puts together all previous language-related knowledge establishing hierarchy of language families. Finally, in Chapter 6 we establish the main results of this thesis and explain their validity in detail. We present simple tree-based conditions under which general, regular-controlled context-free, and scattered context grammars and cooperating distributed grammar systems generate context-free languages. After that, Chapter 7 shows how to apply the main results in practice for proving context-freeness of languages. The thesis is closed by stating several open problems rising from the subject of this work and by suggesting some future research perspectives.

Chapter 2

Languages and Their Models

In this introductory chapter, we define all the basic notions concerning formal language theory which is necessary to properly understand and follow the main matter of this work, however, if the reader is already familiar with the basics of formal languages, the chapter may be skipped and treated as a reference for terminology. We assume that the reader is familiar with discrete mathematics including set theory (see [2, 9, 21]). Some of the following definitions and notions were introduced in [14, 43].

2.1 Languages

For a set W, $\mathrm{card}(W)$ denotes its *cardinality*. An *alphabet* Σ is a finite, nonempty set of elements called *symbols*. If $\mathrm{card}(\Sigma) = 1$, then Σ is a *unary alphabet*. A *string* or, synonymously, a *word* over Σ is any finite sequence of symbols from Σ. We omit all separating commas in strings; that is, for a string $a_1, a_2, \dots, a_n$, for some $n \geq 1$, we write $a_1 a_2 \cdots a_n$ instead. The *empty string*, denoted by ε, is the string that is formed by no symbols, i.e. the empty sequence. By Σ^*, we denote the *set of all strings* over Σ (including ε). Set $\Sigma^+ = \Sigma^* - \{\varepsilon\}$.

Let x be a string over Σ, i.e. $x \in \Sigma^*$, and express x as $x = a_1 a_2 \cdots a_n$, where $a_i \in \Sigma$, for all $i = 1 \dots, n$, for some $n \geq 0$ (the case when $n = 0$ means that $x = \varepsilon$). The *length* of x, denoted by $|x|$, is defined as $|x| = n$. The *reversal* of x, denoted by $\mathrm{reversal}(x)$, is defined as $\mathrm{reversal}(x) = a_n a_{n-1} \cdots a_1$. The *alphabet* of x, denoted by $\mathrm{alph}(x)$, is defined as $\mathrm{alph}(x) = \{a_1, a_2, \dots, a_n\}$; informally, it is the set of symbols appearing in x. For $U \subseteq \Sigma$, $\#_U(x)$ denotes the number of occurrences of symbols from U in x. If $U = \{a\}$, then instead of $\#_{\{a\}}(x)$, we write just $\#_a(x)$. The *leftmost symbol* of x, denoted by $\mathrm{lms}(x)$, is defined as $\mathrm{lms}(x) = a_1$ if $n \geq 1$ and $\mathrm{lms}(x) = \varepsilon$ otherwise. The *rightmost symbol* of x, denoted by $\mathrm{rms}(x)$, is defined analogously. If $n \geq 1$, then for every $i = 1, \dots, n$, let $\mathrm{sym}(x, i)$ denote the ith symbol in x. Notice that $|\varepsilon| = 0$, $\mathrm{reversal}(\varepsilon) = \varepsilon$, and $\mathrm{alph}(\varepsilon) = \emptyset$,

Let x and y be two strings over Σ. Then, xy is the *concatenation* of x and y. Note that $x\varepsilon = \varepsilon x = x$. If x can be written in the form $x = uv$, for some $u, v \in \Sigma^*$, then u is a *prefix* of x and v is a *suffix* of x. If $0 < |u| < |x|$, then u is a *proper prefix* of x; similarly, if $0 < |v| < |x|$, then v is a *proper suffix* of x. Define $\mathrm{prefix}(x) = \{u \mid u \text{ is a prefix of } x\}$ and $\mathrm{suffix}(x) = \{v \mid v \text{ is a suffix of } x\}$. For every $i \geq 0$, $\mathrm{prefix}(x, i)$ is the prefix of x of length i if $|x| \geq i$, and $\mathrm{prefix}(x, i) = x$ if $|x| < i$. If $x = uvw$, for some $u, v, w \in \Sigma^*$, then v is a *substring* of x. The set of all substrings of x is denoted by $\mathrm{sub}(x)$. Moreover,

$$\mathrm{sub}(y, k) = \big\{x \mid x \in \mathrm{sub}(y), |x| \leq k\big\}$$

Let n be a nonnegative integer. Then, the *nth power* of x, denoted by x^n, is a string over Σ recursively defined as

$$\begin{array}{ll}(1) & x^0 = \varepsilon \\ (2) & x^n = xx^{n-1} \text{ for } n \geq 1\end{array}$$

Let $x = a_1a_2 \cdots a_n$ be a string over Σ, for some $n \geq 0$. The set of all *permutations* of x, denoted by $\mathrm{perm}(x)$, is defined as

$$\begin{array}{rcl}\mathrm{perm}(x) & = & \big\{b_1b_2 \cdots b_n \mid b_i \in \mathrm{alph}(x), \text{ for all } i = 1, \ldots, n, \text{ and} \\ & & (b_1, b_2, \ldots, b_n) \text{ is a permutation of } (a_1, a_2, \ldots, a_n)\big\}\end{array}$$

Note that $\mathrm{perm}(\varepsilon) = \{\varepsilon\}$.

A *language* L over Σ is any set of strings over Σ, i.e. $L \subseteq \Sigma^*$. The set Σ^* is called the *universal language* because it consists of all strings over Σ. If L is a finite set, then it is a *finite language*; otherwise, it is an *infinite language*. If $\mathrm{card}(\Sigma) = 1$, then L is a *unary language*. The *empty language* is denoted by $\emptyset$. All the common set operations are also applicable on languages.

The *alphabet* of L, denoted by $\mathrm{alph}(L)$, is defined as

$$\mathrm{alph}(L) = \bigcup_{x \in L} \mathrm{alph}(x)$$

The *permutation* of L, denoted by $\mathrm{perm}(L)$, is defined as

$$\mathrm{perm}(L) = \big\{\mathrm{perm}(x) \mid x \in L\big\}$$

The *reversal* of L, denoted by $\mathrm{reversal}(L)$, is defined as

$$\mathrm{reversal}(L) = \big\{\mathrm{reversal}(x) \mid x \in L\big\}$$

As all languages are sets, all common operations over sets can be applied to them. There are also some special operations which apply only to languages. The *concatenation* of L_1 and L_2, denoted by L_1L_2, is the set

$$L_1L_2 = \big\{x_1x_2 \mid x_1 \in L_1 \text{ and } x_2 \in L_2\big\}$$

Note that $L\{\varepsilon\} = \{\varepsilon\}L = L$. For $n \geq 0$, the *nth power* of L, denoted by L^n, is recursively defined as

$$\begin{array}{ll}(1) & L^0 = \{\varepsilon\} \\ (2) & L^n = L^{n-1}L\end{array}$$

Let Σ be an alphabet. For $x, y \in \Sigma^*$, the *shuffle* of x and y, denoted by $\mathrm{shuffle}(x, y)$, is defined as

$$\begin{array}{rcl}\mathrm{shuffle}(x, y) & = & \big\{x_1y_1x_2y_2 \cdots x_ny_n \mid x = x_1x_2 \ldots x_n, y = y_1y_2 \cdots y_n, \\ & & x_i, y_i \in \Sigma^*, 1 \leq i \leq n, n \geq 1\big\}\end{array}$$

Let Σ and Γ be two alphabets. Let K and L be languages over alphabets Σ and Γ, respectively. A total function σ from Σ^* to 2^{Γ^*} such that $\sigma(uv) = \sigma(u)\sigma(v)$, for every $u, v \in \Sigma^*$, is a *substitution*. A substitution is *ε-free* if it is defined from Σ^* to 2^{Γ^+}. If $\sigma(a)$ for every $a \in \Sigma$ is finite, then σ is said to be *finite*. By this definition, $\sigma(\varepsilon) = \{\varepsilon\}$ and $\sigma(a_1a_2 \cdots a_n) = \sigma(a_1)\sigma(a_2) \cdots \sigma(a_n)$, where $n \geq 1$ and $a_i \in \Sigma$, for all $i = 1, 2, \ldots, n$, so σ is completely specified by defining $\sigma(a)$ for each $a \in \Sigma$. For $L \subseteq \Sigma^*$, we extend the definition of σ to

$$\sigma(L) = \bigcup_{w \in L} \sigma(w)$$

A total function φ from Σ^* to Γ^* such that $\varphi(uv) = \varphi(u)\varphi(v)$, for every $u, v \in \Sigma^*$, is a *homomorphism* or, synonymously, a *morphism*. As any homomorphism is a special case of finite substitution, we specify φ by analogy with the specification of σ. For $L \subseteq \Sigma^*$, we extend the definition of φ to

$$\varphi(L) = \{\varphi(w) \mid w \in L\}$$

By analogy with substitution, φ is *ε-free* if $\varphi(a) \neq \varepsilon$, for every $a \in \Sigma$. By φ^{-1}, we denote the *inverse homomorphism*, defined as

$$\varphi^{-1}(u) = \{w \mid \varphi(u) = w\}$$

By analogy with set theory, sets whose members are languages are called *families of languages*. We define them by describing their properties as follows or by introducing formal models which specify their respective strings as shown in Sections 2.2 and 2.3.

Definition 2.1.1. *For an alphabet Σ and a positive integer k, a lagauge L consisted of finitely many strings $w_1, w_2, \ldots, w_k \in \Sigma^*$ is called* finite language. *The family of finite languages is denoted by* **FIN**. □

Definition 2.1.2. *A language L over an alphabet Σ is called regular if it can be constructed by a finite number of applications of the operations union, concatenation, and power from subsets of $\Sigma \cup \{\varepsilon\}$. By* **REG** *we denote the family of all regular languages.* □

Let τ be a k-ary operation on languages and let $\mathcal{L}$ be a family of languages. We say that $\mathcal{L}$ is closed under the operation τ, if, for all languages L_1, L_2, $\ldots$, $L_k \in \mathcal{L}$, $\tau(L_1$, L_2, $\ldots$, $L_k)$ is also an element of $\mathcal{L}$. A language family is called an *abstract family of languages* (abbreviated AFL) if it is closed under union, concatenation, positive power, ε-free homomorphism, inverse homomorphism, and intersection with regular language. An AFL is called full if it is closed with respect to an arbitrary homomorphism. A family of languages closed under all AFL operations except concatenation and positive iteration is termed a *semi-AF*; a semi-AFL is full if it is closed under an arbitrary homomorphism.

2.2 Grammars

In this section, we define language-generating devices called grammars which play a major role in formal language theory as well as in this work.

Definition 2.2.1. *A* general grammar[1] *(GG for short) is a quadruple*

$$G = (V, T, P, S)$$

where V is a total alphabet*, $T \subset V$ is an alphabet of* terminals*, $N = V - T$ is an alphabet of* nonterminals*, P is a finite relation from V^*NV^* to V^*, $S \in N$ is the* start symbol.

Pairs $(u, v) \in P$ are called rewriting rules *(abbreviated* rules*), and are written as $u \to v$. A rewriting rule $u \to v \in P$ satisfying $v = \varepsilon$ is called an* erasing rule. *If there is no such*

[1] also referred as phrase-structure grammar

rule in P, then we say that G is a propagating *grammar. A rule $u \to v \in P$ is called* context-free *if u consists of a single nonterminal, otherwise, it is a* non-context-free *rule. Define the* domain of P *as* $\mathrm{dom}(P) = \{A \mid A \to x \in P\}$.

The G-based direct derivation relation *over V^* is denoted by $\Rightarrow_G$ and defined as*

$$x \Rightarrow_G y$$

if and only if $x = x_1ux_2, y = x_1vx_2$, and $u \to v \in P$, where $x_1, x_2 \in V^*$.

Since $\Rightarrow_G$ is a relation, $\Rightarrow_G^k$ is the kth power of $\Rightarrow_G$, for $k \geq 0$, $\Rightarrow_G^+$ is the transitive closure of $\Rightarrow_G$, and $\Rightarrow_G^*$ is the reflexive-transitive closure of $\Rightarrow_G$. Let $D\colon S \Rightarrow_G^* x$ be a derivation, for some $x \in V^*$. Then, x is a sentential form*. If $x \in T^*$, then x is a* sentence*. If x is a sentence, then D is a* terminal derivation.

The language *of G, denoted by* $\mathrm{L}(G)$*, is the set of all sentences defined as*

$$\mathrm{L}(G) = \{w \in T^* \mid S \Rightarrow_G^* w\}$$ □

Next, for every general grammar G, we define two sets, $\mathscr{F}(G)$ and $\mathscr{T}(G)$. $\mathscr{F}(G)$ contains all sentential forms of G. $\mathscr{T}(G)$ contains all sentential forms from which there is a derivation of a string in $\mathrm{L}(G)$. The notion also applies for other later defined types of grammars.

Definition 2.2.2. *Let $G = (V, T, P, S)$ be a general grammar. Then,*

$$\mathscr{F}(G) = \{x \in V^* \mid S \Rightarrow_G^* x\}$$

is the set of all sentential forms of G *and*

$$\mathscr{T}(G) = \{x \in \mathscr{F}(G) \mid x \Rightarrow_G^* y,\ y \in T^*\}$$

is the set of all sentential forms from which there is a derivation of a string in $\mathrm{L}(G)$*.* □

For brevity, we often denote a rule $u \to v$ with a unique label r as $r\colon u \to v$, and instead of $u \to v \in P$, we simply write $r \in P$. The notion of rule labels is formalized in the following definition.

Definition 2.2.3. *Let $G = (V, T, P, S)$ be a general grammar. Let Ψ be a set of symbols called* rule labels *such that* $\mathrm{card}(\Psi) = \mathrm{card}(P)$*, and ψ be a bijection from P to Ψ. For simplicity and brevity, to express that ψ maps a rule, $u \to v \in P$, to r, where $r \in \Psi$, we write $r\colon u \to v \in P$; in other words, $r\colon u \to v$ means that $\psi(u \to v) = r$. For $r\colon u \to v \in P$, u and v represent the* left-hand side *of r, denoted by* $\mathrm{lhs}(r)$*, and the* right-hand side *of r, denoted by* $\mathrm{rhs}(r)$*, respectively. We extend ψ from P to P^* in the following way*

(1) $\psi(\varepsilon) = \varepsilon$
(2) $\psi(r_1r_2 \cdots r_n) = \psi(r_1)\psi(r_2) \cdots \psi(r_n)$

for any sequence of rules $r_1r_2 \cdots r_n$, where $r_i \in P$, for all $i = 1, 2, \dots, n$, for some $n \geq 1$.

Let $w_0, w_1, \dots, w_n$ be a sequence of strings, where $w_i \in V^*$, for all $i = 0, 1, \dots, n$, for some $n \geq 1$. If $w_{j-1} \Rightarrow_G w_j$ according to r_j, where $r_j \in P$, for all $j = 1, 2, \dots, n$, then we write

$$w_0 \Rightarrow_G^n w_n\ [r_1r_2 \cdots r_n]$$

For any string w, we write

$$w \Rightarrow_G^0 w\ [\varepsilon]$$

For any two strings w and y, if $w \Rightarrow_G^n y\ [\varrho]$ for $n \geq 0$ and $\varrho \in \Psi^*$, then we write

$$w \Rightarrow_G^* y\ [\varrho]$$

If $n \geq 1$, which means that $|\varrho| \geq 1$, then we write

$$w \Rightarrow_G^+ y\ [\varrho]$$

If $w = S$, then ϱ is called the sequence of rules (rule labels) *used in the derivation of y.* □

In what follows, for any grammar G, we automatically assume that V, N, T, P, S, and Ψ denote its total alphabet, the alphabet of nonterminals, the alphabet of terminals, the set of rules, the start symbol, and the set of rule labels, respectively.

Definition 2.2.4. *A* recursively enumerable language *is a language generated by a general grammar. The family of recursively enumerable languages is denoted by* **RE**. □

Definition 2.2.5. *A general grammar $G = (V, T, P, S)$ is* monotone *if $x \rightarrow y \in P$ implies $|x| \leq |y|$; additionally, if $\varepsilon \in \mathrm{L}(G)$, $S \rightarrow \varepsilon \in P$ and $S \notin \mathrm{alph}(y)$.* □

Definition 2.2.6. *A* context-sensitive grammar *is a general grammar $G = \big(V, T, P, S\big)$ such that every $u \rightarrow v$ in P is of the form*

$$u = x_1 A x_2,\ v = x_1 y x_2$$

where $x_1, x_2 \in V^*$, $A \in N$, and $y \in V^+$. A context-sensitive language *is a language generated by a context-sensitive grammar. The family of context-sensitive languages is denoted by* **CS**. □

Theorem 2.2.1 (see [48]). *The families of languages generated by monotone and context-sensitive grammars are equivalent.*

Definition 2.2.7. *A* context-free grammar *is a general grammar which has only context-free rules. A* context-free language *is a language generated by a context-free grammar. The family of context-free languages is denoted by* **CF**. *Propagating CFGs characterize* **CF** *as well (see [48]).* □

Definition 2.2.8. *Context-free grammar $G = (V, T, P, S)$ is of* index k *(${}_k$CFG for short), for some $k \geq 1$, if for every $w \in \mathrm{L}(G)$ there exists a derivation*

$$S \Rightarrow x_1 \Rightarrow x_2 \Rightarrow \cdots \Rightarrow x_n \Rightarrow w$$

where $\#_N(x_i) \leq k$, for all $1 \leq i \leq n$, for some $n \geq 0$. L is a context-free language of index k *if there exists a ${}_k$CFG G, where $\mathrm{L}(G) = L$. The* family of all context-free languages of index k *generated by ${}_k$CFGs is denoted by ${}_k$***CF**. □

Definition 2.2.9. *A* linear grammar *is a general grammar $G = \big(V, T, P, S\big)$ such that every rule in P is of the form*

$$A \rightarrow xBy \text{ or } A \rightarrow x$$

where $A, B \in N$ and $x, y \in T^*$. A linear language *is a language generated by a linear grammar. The family of linear languages is denoted by* **LIN**. □

Definition 2.2.10. *A* regular grammar *is a general grammar $G = \big(V, T, P, S\big)$ such that every rule in P is of the form*

$$A \to aB \text{ or } A \to a$$

where $A, B \in N$ and $a \in T$. A regular language *is a language generated by a regular grammar. The family of regular languages is denoted by* **REG**. □

Next, we define two *regulated grammars* which generates their sentences by context-free grammars regulated by additional control mechanisms.

Definition 1. A *matrix grammar* (MG for short) is a pair $H = (G, M)$, where

- $G = (V, T, P, S)$ is a context-free grammar;
- M is a finite language over the alphabet of rules ($M \subseteq P^*$).

For $x, y \in V^*$, $m \in M$, H performs a *direct derivation step* from x to y according to the *matrix* m denoted by $x \Rightarrow_H y\ [m]$, if and only if there are x_0, x_1, …, x_n such that $x_0 = x$, $x_n = y$, and

- $x_0 \Rightarrow_G x_1\ [p_1] \Rightarrow_G x_2\ [p_2] \Rightarrow_G \dots \Rightarrow_G x_n\ [p_n]$, and
- $m = p_1 p_2 \dots p_n$, where $p_i \in P$, $1 \leq i \leq n$, for some $n \geq 1$.

The transitive and reflexive closure is defined and denoted as usual. Then,

$$\mathscr{L}(H) = \{x \in T^* \mid S \Rightarrow_H^* x\}$$

is the *language generated by H*. Let **MT** denotes the *family of languages generated by matrix grammars*; the *family of matrix languages* introduced in [1].

Definition 2.2.11. *A* regular-controlled grammar *(an RCG for short) H is a pair $H = (G, C)$, where*

- core grammar $G = (V, T, P, S)$ *is a context-free grammar;*
- control language $C \subseteq P^*$ *is a regular language.*

If $S \Rightarrow_G^* w\ [\alpha]$ and $\alpha\beta \in C$, for some $\alpha, \beta \in P^*$, we put $S \Rightarrow_H^* w\ [\alpha]$ or $S \Rightarrow_H^* w$ for short. The language generated by H*, denoted by* $\mathrm{L}(H)$*, is defined as*

$$\mathrm{L}(H) = \{w \in T^* \mid S \Rightarrow_G^* w\ [\alpha], \alpha \in C\}$$

The family of all regular-controlled languages *is denoted by* **RC**. □

Let us demonstrate the notion of regular-controlled grammars.

Example 2.2.1. Let $H = (G, C)$ be an RCG with $G = (\{S, A, B\}, \{a, b\}, P, S)$, where

$$\begin{aligned} P = \{\quad & 1\colon S \to AB, \\ & 2\colon A \to aA, \quad 3\colon B \to aB, \\ & 4\colon A \to bA, \quad 5\colon B \to bB, \\ & 6\colon A \to a, \quad 7\colon B \to a, \\ & 8\colon A \to b, \quad 9\colon B \to b \quad \} \end{aligned}$$

and

$$C = 1\{23, 45\}^*\{67, 89\}.$$

Observe the control language. After applying the initial rule, there are always consecutive pairs of rules 23 and 45 applied. Eventually, the derivation finishes with application of rules 67 or 89. Then,

$$L(H) = \{ww \mid w \in \{a, b\}^+\}$$

which is the well-known non-context-free context-sensitive language. □

Example 2.2.2. Let $H = (G, C)$ be an RCG with $G = (\{S, A, B\}, \{a, b, c, d\}, P, S)$, where

$$\begin{array}{lll} P = \{ & 1\colon S \to aAcB, & \\ & 2\colon A \to aA, & 3\colon B \to cB, \\ & 4\colon A \to bA, & 5\colon B \to dB, \\ & 6\colon A \to b, & 7\colon B \to d \quad \} \end{array}$$

and

$$C = 1\{23\}^*\{45\}^*67.$$

Observe the construction of H. By the control language, after the application of the initial rule, first, zero or more consecutive applications of rules 23 are performed, second, zero or more consecutive applications of rules 45 are performed. Finally, the derivation finishes by the rules 67. Then,

$$L(H) = \{a^m b^n c^m d^n \mid m, n \geq 1\}$$

which is the well-known non-context-free context-sensitive language. □

Next, we define the notion of scattered context grammars.

Definition 2.2.12. *A* scattered context grammar *(SCG for short) G is a quadruple $G = (V, T, P, S)$, where V, N, T, S, and Ψ have the same meaning as in the case of general grammars and*

$$P \subseteq \bigcup_{m=1}^{\infty} N^m \times (V^*)^m$$

is a finite set of scattered context rules. *Instead of $p\colon (A_1, A_2, \ldots, A_n, x_1, x_2, \ldots, x_n) \in P$, where $p \in \Psi$, $A_i \in N$, $x_i \in V^*$, for $1 \leq i \leq n$, for some $n \geq 1$, we write $p\colon (A_1, A_2, \ldots, A_n) \to (x_1, x_2, \ldots, x_n)$. If*

$$u = u_0 A_1 u_1 A_2 \ldots u_{n-1} A_n u_n$$
$$v = u_0 x_1 u_1 x_2 \ldots u_{n-1} x_n u_n$$

and $p\colon (A_1, A_2, \ldots, A_n) \to (x_1, x_2, \ldots, x_n) \in P$, where $u_i \in V^*$, $0 \leq i \leq n$, then G makes a derivation step *from u to v according to p, symbolically written as $u \Rightarrow_G v\ [p]$ or, simply, $u \Rightarrow_G v$. Set $\mathrm{len}(p) = n$. If $\mathrm{len}(p) \geq 2$, p is said to be a* non-context-free rule *while for $\mathrm{len}(p) = 1$, p is said to be* context-free. *Define $\Rightarrow^k_G$, $\Rightarrow^*_G$, and $\Rightarrow^+_G$ in the standard way.*

The language *of G is $\mathrm{L}(G) = \{w \in T^* \mid w \in \mathscr{F}(G)\}$. A language L is a* scattered context language *if there exists a scattered context grammar G such that $L = \mathrm{L}(G)$. The family of scattered context languages coincide with* **RE** *(see [40]).* □

Notice that if for a scattered context grammar G, every rule p is of $\mathrm{len}(p) = 1$, G is in fact a context-free grammar.

Finally, we define cooperating distributed grammar systems.

Definition 2.2.13. *Let $n \geq 1$ be a positive integer. A* cooperating distributed grammar system *(CDGS for short) of degree n is an $(n+3)$-tuple*

$$\Gamma = (V, T, S, P_1, \ldots, P_n),$$

where V, T, S are defined as in Definition 2.2.1 and P_i is a finite set of context-free rules, called component of Γ, *for $i = 1, \ldots, n$; then, $G_i = (V, T, P_i, S)$ is a context-free grammar.*

Let $u, v \in V^*$. Then,

$$u \Rightarrow^{\leq k}_{\Gamma(i)} v, \; u \Rightarrow^{=k}_{\Gamma(i)} v, \; u \Rightarrow^{\geq k}_{\Gamma(i)} v, \; u \Rightarrow^{*}_{\Gamma(i)} v$$

iff

$$u \Rightarrow^{l}_{G_i} v, \; u \Rightarrow^{k}_{G_i} v, \; u \Rightarrow^{m}_{G_i} v, \; u \Rightarrow^{*}_{G_i} v$$

respectively, for some $i = 1, \ldots, n$ and $1 \leq l \leq k \leq m$. Additionally, $u \Rightarrow^{t}_{\Gamma(i)} v$ iff $u \Rightarrow^{*}_{G_i} v$ and $\mathrm{dom}(P_i) \cap \mathrm{alph}(v) = \emptyset$. Let $\psi \in \{*, t\} \cup \{\leq k, =k, \geq k \mid k \geq 1\}$. Then, ψ is called mode of derivation. *The language generated in ψ-mode by a CDGS G of degree n is defined as*

$$\begin{aligned} \mathrm{L}(G^\psi) = \{w \in T^* \mid S \Rightarrow^{\psi}_{\Gamma(i_1)} w_1 \Rightarrow^{\psi}_{\Gamma(i_2)} w_2 \Rightarrow^{\psi}_{\Gamma(i_3)} \cdots \Rightarrow^{\psi}_{\Gamma(i_m)} w_m, \\ 1 \leq i_j \leq n, 1 \leq j \leq m, m \geq 1, w_m = w\}. \end{aligned}$$

Let $n \geq 1$ be an integer and ψ be a mode of derivation. By $\mathbf{CD}^{\psi}_{n}$ we denote the family of all languages L, for which there is a CDGS G of degree n such that $\mathrm{L}(G^\psi) = L$. □

Theorem 2.2.2. (Csuhaj-Varjú et al. [12]).

1. *Let $\psi \in \{*, =1, \geq 1\} \cup \{\leq k \mid k \geq 1\}$. For $n \geq 1$, $\mathbf{CD}^{\psi}_{n} = \mathbf{CF}$.*

2. *Let $\psi \in \{=k, \geq k \mid k \geq 2\}$. For $n \geq 3$,*

$$\mathbf{CF} = \mathbf{CD}^{\psi}_{1} \subset \mathbf{CD}^{\psi}_{2} \subseteq \mathbf{CD}^{\psi}_{n} \subseteq \mathbf{CD}^{\psi}_{n+1} \subseteq \mathbf{MT},$$

   where **MT** *is the* family of matrix languages.

3. *For $n \geq 3$,*

$$\mathbf{CF} = \mathbf{CD}^{t}_{1} = \mathbf{CD}^{t}_{2} \subset \mathbf{CD}^{t}_{3} = \mathbf{CD}^{t}_{n} = \mathbf{ET0L},$$

   where **ET0L** *is the* family of languages generated by extended tabled zero-sided Lindenmayer systems.

For definitions and properties of **MT** *and* **ET0L** *see [43].*

Normal Forms

In this section, we convert previously introduced grammars into several normal forms.

Definition 2.2.14. *Let $G = (V, T, P, S)$ be a general grammar. G is in the* Kuroda normal form *(see [27]) if every rule in P is of one of the following four forms*

$$(i)\; AB \to CD \qquad (ii)\; A \to BC \qquad (iii)\; A \to a \qquad (iv)\; A \to \varepsilon$$

where $A, B, C, D \in N$, and $a \in T$. Additionally, if every rule in P is of one of the forms (i) through (iii), G is a monotone general grammar in the Kuroda normal form. □

Theorem 2.2.3 (see [27]). *A language L is recursively enumerable iff $L = \mathrm{L}(G)$, where G is a general grammar in the Kuroda normal form.*

Theorem 2.2.4 (see [27]). *A language L is context-sensitive iff $L = \mathrm{L}(G)$, where G is a monotone general grammar in the Kuroda normal form.*

Definition 2.2.15. *Let $G = (V, T, P, S)$ be a general grammar. G is in the* binary form *if any $p \in P$ has one of the following three forms*

$$(i)\ AB \to CD \qquad (ii)\ A \to BC \qquad (iii)\ A \to X$$

where $A, B, C, D \in N$, $X \in V \cup \{\varepsilon\}$. □

Theorem 2.2.5. *A language L is recursively enumerable iff $L = \mathrm{L}(G)$, where G is a general grammar in the binary form.*

Proof. Let $G = (V, T, P, S)$ be a general grammar in the binary form. Notice that if G is not in the Kuroda normal form, P contains rules of the form $A \to B$, where $A, B \in N$. Construct $G' = (V, T, P', S)$ as follows. Put all rules from P which satisfy Kuroda normal form to P'. For every $A, B \in N$, where $A \Rightarrow_G^* B$,

1. for every $r\colon B \to u \in P$, where r satisfy Kuroda normal form, introduce new rule $A \to u$ into P';

2. for every $r\colon XB \to YZ \in P$, introduce new rule $XA \to YZ$ into P';

3. for every $r\colon BX \to YZ \in P$, introduce new rule $AX \to YZ$ into P'.

Obviously, $\mathrm{L}(G) = \mathrm{L}(G')$ and G' is in the Kuroda normal form. □

Theorem 2.2.6. *A language L is context-sensitive iff $L = \mathrm{L}(G)$, where G is a monotone general grammar in the binary form.*

Proof. Prove by analogy with Theorem 2.2.5. □

Definition 2.2.16. *Let $G = (V, T, P, S)$ be a context-free grammar. G is in the* Chomsky normal form *(see [6]) if any $p \in P$ has one of these forms,*

$$(i)\ A \to BC \qquad (ii)\ A \to a \qquad (iii)\ S \to \varepsilon$$

where $A, B, C \in N$ and $a \in T$. □

Theorem 2.2.7 (see [6]). *A language L is context-free iff $L = \mathrm{L}(G)$, where G is a context-free grammar in the Chomsky normal form.*

Definition 2.2.17. *Let $G = (V, T, P, S)$ be a context-free grammar. G is in the* binary form *if any $p \in P$ has one of the following two forms*

$$(i)\ A \to BC \qquad (ii)\ A \to X$$

where $A, B, C \in N$, $X \in V^*$, and $\#_N(X) \leq 1$. □

Theorem 2.2.8. *A language L is context-free iff $L = \mathrm{L}(G)$, where G is a context-free grammar in the binary form.*

Proof. Since the binary form of context-free grammars is a generalized Chomsky normal form and every context-free grammar in Chomsky normal form is also in the binary form, Theorem 2.2.8 holds. □

Definition 2.2.18. *A regular-controlled grammar is in the* binary form *if its core grammar is in the binary form.* □

Theorem 2.2.9. *Let $H = (G, C)$ be an arbitrary regular-controlled grammar. Then, there exists a regular-controlled grammar $H' = (G', C')$ in the binary form, where $\mathrm{L}(H) = \mathrm{L}(H')$.*

Proof. We introduce Algorithm 1 for conversion of a regular-controlled grammar into a corresponding regular-controlled grammar in the binary form and prove its correctness.

Algorithm 1 Conversion of RCG into the binary form

```
Input: An arbitrary RCG $H = (G, C)$, $G = (V, T, P, S)$.
Output: RCG $H'$ in the binary form with $\mathrm{L}(H') = \mathrm{L}(H)$.
 1: Construct $H' = (G', C')$, $G' = (V', T, P', S)$; $C' = P' = \emptyset$, $V' = V$.
 2: for all $r \in P$ do
 3:   if $r$ satisfies the binary form then
 4:     $P' \leftarrow P' \cup \{r\}$
 5:     $P \leftarrow P - \{r\}$
 6:   end if
 7: end for
 8: $i \leftarrow 0$
 9: while there exists $r\colon A \to w \in P$ do
10:   for $uXv = w$ and $\mathrm{alph}(u) \cap N = \emptyset$ and $X \in N$ do
11:     $N' \leftarrow \langle uX \rangle, \langle v \rangle$
12:     $P' \leftarrow P' \cup \{r_1\colon A \to \langle uX \rangle\langle v \rangle, r_2\colon \langle uX \rangle \to uX\}$
13:     if $\langle v \rangle \to v$ satisfies the binary form then
14:       $P' \leftarrow P' \cup \{r_3\colon \langle v \rangle \to v\}$
15:     else
16:       $P \leftarrow P \cup \{r_3\colon \langle v \rangle \to v\}$
17:     end if
18:   end for
19:   Define a new homomorphism $h_i$ over $P \cup P'$:
20:      $h_i(x) = r_1 r_2 r_3$, for $x = r$;
21:      $h_i(x) = x$, otherwise.
22:   $P \leftarrow P - \{r\}$
23:   $i \leftarrow i + 1$
24: end while // $P = \emptyset$
25: $C' \leftarrow \{w \mid w = h_i(h_{i-1}(\cdots h_1(h_0(x))\cdots)), x \in C\}$
```

Claim 1. Algorithm 1 is correct.

Proof. Basic idea. The initial steps 1 through 8 construct a template for the resulting grammar which enters the following iterative process.

During 9 through 24 every rule which does not satisfy binary form is decomposed and replaced by three new rules. The first and the second rule is in the binary form. If the third rule does not satisfy the binary form, it enters the following iterative process, however, with shorter right-hand side than the replaced one; this ensures the finiteness of the procedure. Additionally, a new homomorphism is introduced to substitute the replaced rule in the control language with the sequence of newly introduced rules. This iterative process finally defines a finite hierarchy of homomorphisms.

Since **REG** is closed under homomorphism (see [50]), in the last 25th step of the algorithm a new control language is established by application of the defined homomorphisms. The complete rigorous proof is left to the reader. $\square$

Since for any RCG H we can construct an RCG H' in the binary form, where $\mathrm{L}(H) = \mathrm{L}(H')$, Theorem 2.2.9 holds. $\square$

Definition 2.2.19. *Let $G = (V, T, P, S)$ is a scattered context grammar. G is in the* binary form *if any $p \in P$ has one of the following three forms*

$$\textit{(i)}\ (A, B) \rightarrow (C, D) \qquad \textit{(ii)}\ (A) \rightarrow (BC) \qquad \textit{(iii)}\ (A) \rightarrow (X)$$

where $A, B, C, D \in N$, $X \in V \cup \{\varepsilon\}$. $\square$

Theorem 2.2.10. *A language L is recursively enumerable iff $L = \mathrm{L}(G)$, where G is a scattered context grammar in the binary form.*

Proof. Since the binary form is generalized version of 2-limited form of scattered context grammars from [43], theorem holds. For details see Theorem 4.7.11. in [43].

The following example introduces propagating scattered context grammar in the binary form which generates a non-contetx-free context-sensitive language.

Example 2.2.3. Let $G = (V, T, P, S)$ be an SCG, where

$$V = \{S, A, B, C, \bar{A}, \bar{B}, \bar{a}, \bar{b}, \bar{c}, a, b, c\},$$

$T = \{a, b, c\}$, and

$$\begin{array}{rclll}
P & = & \{ & 1\colon (S) \rightarrow (AB), & 6\colon (A) \rightarrow (a), \\
 & & & 2\colon (A, B) \rightarrow (\bar{A}, \bar{B}), & 7\colon (B) \rightarrow (\bar{b}\bar{c}), \\
 & & & 3\colon (\bar{A}) \rightarrow (\bar{a}A), & 8\colon (\bar{a}) \rightarrow (a), \\
 & & & 4\colon (\bar{B}) \rightarrow (\bar{b}C), & 9\colon (\bar{b}) \rightarrow (b), \\
 & & & 5\colon (C) \rightarrow (B\bar{c}), & 10\colon (\bar{c}) \rightarrow (c)\ \}.
\end{array}$$

G is obviously propagating SCG in the binary form. Observe the rules of G. Initially, rule 1 generates AB. The derivation may possibly finish by the rules 6 and 7 or continue by the rule 2. After the rule 2, rules 3, 4, and 5 must be applied. Without any loss of generality, suppose that rules 8, 9, and 10 are applied only at the very end of every derivation. Then,

$$\bar{a}^k A \bar{b}^k B \bar{c}^k \Rightarrow_G^3 \bar{a}^{k+1} A \bar{b}^{k+1} B \bar{c}^{k+1}\ [345],$$

for $k \geq 0$. For example $aaabbbccc$ is generated as follows.

$$\begin{array}{llllll}
S & \Rightarrow_G & AB & [1] & \Rightarrow_G & \bar{A}\bar{B} & [2] \\
 & \Rightarrow_G & \bar{a}A\bar{B} & [3] & \Rightarrow_G & \bar{a}A\bar{b}C & [4] \\
 & \Rightarrow_G & \bar{a}A\bar{b}B\bar{c} & [5] & \Rightarrow_G & \bar{a}\bar{A}\bar{b}\bar{B}\bar{c} & [2] \\
 & \Rightarrow_G & \bar{a}\bar{a}A\bar{b}\bar{B}\bar{c} & [3] & \Rightarrow_G & \bar{a}\bar{a}A\bar{b}\bar{b}C\bar{c} & [4] \\
 & \Rightarrow_G & \bar{a}\bar{a}A\bar{b}\bar{b}B\bar{c}\bar{c} & [5] & \Rightarrow_G & \bar{a}\bar{a}a\bar{b}\bar{b}B\bar{c}\bar{c} & [6] \\
 & \Rightarrow_G & \bar{a}\bar{a}a\bar{b}\bar{b}\bar{b}\bar{c}\bar{c}\bar{c} & [7] & \Rightarrow_G^8 & aaabbbccc & [8\,8\,9\,9\,9\,10\,10\,10]
\end{array}$$

Clearly, $\mathrm{L}(G) = \{a^k b^k c^k \mid k \geq 1\}$ which is the well-known non-context-free context sensitive language—$\mathrm{L}(G) \in$ **CS** $-$ **CF** (see Chapter 5 for the proof). $\square$

2.3 Automata

Language accepting devices called automata stay as the essential counterparts to grammars. However, we do not cover them in an exhaustive way, since in this work they represent only a minor matter.

Definition 2.3.1. *A* finite automaton *(an FA for short) is a quintuple* $M = (Q, \Sigma, R, s, F)$*, where* Q *is a finite set of* states, Σ *is an* input alphabet*, where* $Q \cap \Sigma = \emptyset$, $R \subseteq Q \times (\Sigma \cup \{\varepsilon\}) \times Q$ *is a finite relation, called the set of* transitions, $s \in Q$ *is the* initial state, $F \subseteq Q$ *is the set of* final states.

Instead of $(p, a, q) \in R$*, we write* $pa \to q \in R$*. A* configuration *of* M *is any word from* $Q\Sigma^*$*. The relation of a* direct move*, denoted by* $\vdash$*, is defined over* $Q\Sigma^*$ *as follows: if* pax, $qx \in Q\Sigma^*$*, and* $pa \to q \in R$*, then* $pax \vdash qx$ *in* M*.*

Let $\vdash^k$, $\vdash^*$*, and* $\vdash^+$ *denote the* k*th power of* $\vdash$*, for some* $k \geq 0$*, the reflexive and transitive closure of* $\vdash$*, and the transitive closure of* $\vdash$*, respectively. The* language accepted by M *is denoted by* $\mathrm{L}(M)$ *and defined as*

$$\mathrm{L}(M) = \big\{w \in \Sigma^* \mid sw \vdash^* f,\ f \in F\big\}.$$

As it is well-known, the family of finite automata describes **REG** *(see [35]).* □

Next, we define two special variants of finite automata.

Definition 2.3.2. *Let* $M = (Q, \Sigma, R, s, F)$ *be a finite automaton.* M *is said to be* deterministic *(DFA for short) if and only if* $pa \to q \in R$ *implies that* $a \neq \varepsilon$ *and* $pa \to q_1, pa \to q_2 \in R$ *implies that* $q_1 = q_2$*, for all* $p, q, q_1, q_2 \in Q$ *and* $a \in \Sigma$*.* M *is said to be* complete *if and only if* M *is deterministic and for all* $p \in Q$ *and all* $a \in \Sigma$, $pa \to q \in R$ *for some* $q \in Q$*.* □

Deterministic finite automata and complete DFAs characterize **REG** as well as finite automata do and every FA can be converted into an equivalent DFA or complete DFA (see [22, 35, 51]).

Chapter 3

Graph-Based Representation of Derivation

This chapter introduces how to represent hierarchical structure of a derivation in the sense of rewritten symbols based on graphs; more precisely, tree structures known as derivation trees. Every symbol introduced by a grammar during the derivation of a sentence corresponds to a node of some derivation tree. All preceding nodes represent symbols which led to introduction of the node in question primarily with the start symbol as the root node, while all successor nodes represent symbols which the node is rewritten to with terminal symbols as the leaf nodes. This representation is essential for the main focus of this work. We later describe the relation between several graph-related properties of derivations of some grammar and the actual power of the grammar. Specifically, we demonstrate that putting some constant restrictions upon these properties of derivations of some non-context-free grammar in fact results into generation of a context-free language.

3.1 Graphs and Trees

In this section, we define all necessary graph-related notions.

Definition 3.1.1. *A* directed graph *G is a pair $G = (V, E)$, where V is a finite* set of nodes*, and $E \subseteq V \times V$ is a finite* set of edges*. For a node $v \in V$, the number of edges of the form $(x, v) \in E$, $x \in V$, is called an* in-degree *of v and denoted by* in-d(v)*. For a node $v \in V$, the number of edges of the form $(v, x) \in E$, $x \in V$, is called an* out-degree *of v and denoted by* out-d(v)*. Let $(v_0, v_1, \dots, v_n)$ be an n-tuple of nodes, for some $n \geq 0$, where $v_i \in V$, for $0 \leq i \leq n$, and there exists an edge $(v_k, v_{k+1}) \in E$, for every pair of nodes v_k, v_{k+1}, where $0 \leq k \leq n-1$, then, we call it a* path of the length *n or simply a* path*. Let $(v_0, v_1, \dots, v_n)$ be a path of the length n, for some $n \geq 0$, where $v_i \neq v_j$, for $0 \leq i \leq n$, $0 \leq j \leq n$, $i \neq j$, then, we call it a* walk*. Let $(v_0, v_1, \dots, v_n)$ be a walk in G, for some $n \geq 0$, except that $v_0 = v_n$, then, we call it a* cycle*. A graph G is* acyclic *if it contains no cycle. A graph G is* connected *if for every pair of nodes $u, v \in V$ there is a path $(v_0, v_1, \dots, v_n)$, for some $n \geq 0$, where $v_0 = u$ and $v_n = v$ or $v_0 = v$ and $v_n = u$; otherwise, it is disconnected. For a graph $G = (V, E)$, a pair $(U, V - U)$, $U \subseteq V$, is a* cut*; then, $G_U = (U, E_U)$ with $E_U = E - \{(u, v) \mid u \notin U \text{ or } v \notin U\}$ is called the* graph generated by *U.* □

Definition 3.1.2. *An* (oriented) tree *is a directed acyclic graph* $t = (V, E)$*, with a specified node* $\hat{r} \in V$ *called the* root *such that* $\text{in-d}(\hat{r}) = 0$*, and for all* $x \in V - \{\hat{r}\}$*,* $\text{in-d}(x) = 1$ *and there exists a path* $(v_0, v_1, \ldots, v_n)$*, where* $v_0 = \hat{r}$*,* $v_n = x$*, for some* $n \geq 1$*. The* depth of t*,* $\text{depth}(t)$*, is the length of the longest path in* t*.*

Let $t = (V, E)$ *be a tree. For* $v, u \in V$*, where* $(v, u) \in E$*,* v *is called a* parent *of* u*,* u *is called a child of* v*, respectively. For* $v, u, z \in V$*, where* $(v, u), (v, z) \in E$*,* u *is called a* sibling *of* z*. A node without any children is called a* leaf*. Define a partial order relation* $<$ *over* V *as follows. For a path* $\alpha = (m_0, m_1, \ldots, m_k)$*, where* $m_0 = \hat{r}$*,* $m_i < m_k$*,* $0 \leq i \leq k-1$*. Then,* m_i *is called a* predecessor *of* m_k *and* m_k *is called a* descendant *of* m_i*. A tree* $t' = (V', E')$ *is a* subtree of t *if* $\emptyset \subset V' \subseteq V$*,* $E' \subseteq E \cap (V' \times V')$*, and in* t*, no node in* $V - V'$ *is a descendant of a node in* V'*;* t' *is an* elementary subtree of t *if* $\text{depth}(t') = 1$*.*

Let $u, v \in V$ *be two nodes, where* $u \neq v$*. Let* $\alpha = (m_0, m_1, \ldots, m_k)$ *and* $\beta = (n_0, n_1, \ldots, n_r)$ *be two paths, where* $m_0 = n_0 = \hat{r}$*,* $m_k = u$*, and* $n_r = v$*, for some* $k, r \geq 0$*. Let* $0 \leq j \leq \min(k, r)$ *be maximal integer such that* $m_i = n_i$*, for all* $0 \leq i \leq j$*. Then, the node* m_j *is called the* lowest common predecessor of u and v*. Note that if* u *is a predecessor of* v *it is also the lowest common predecessor of* u *and* v*.*

An ordered tree t *is a tree, where for every set of siblings there exists a linear ordering. Assume* o *has the children* n_1*,* n_2*,* $\ldots$*,* n_r *ordered in this way, where* $r \geq 1$*. Then,* n_1 *is the* leftmost child *of* o*,* n_r *is the* rightmost child *of* o*,* n_i *is the* direct left sibling *of* n_{i+1}*,* n_{i+1} *is the* direct right sibling *of* n_i*,* $1 \leq i \leq r-1$*, and for* $1 \leq j < k \leq r$*,* n_j *is a* left sibling *of* n_k *and* n_k *is a* right sibling *of* n_j*. Let us extend the ordering according to the transitive closure of parent-children relation. Then, for a tree* t *we have a left-to-right ordered sequence of leafs* l_1*,* l_2*,* $\ldots$*,* l_k*, for some* $k \geq 1$*.*

An ordered tree is called labelled*, if there exists a set of labels* $\mathcal{L}$ *and a total mapping* $l : V \to \mathcal{L}$*. Let* t *be a labelled ordered tree, then the string of labels of all leaves written in the left-to-right order is called the* frontier *of* t *and denoted by* $\text{frontier}(t)$*. In what follows we substitute a node of a tree by its label if there is no risk of confusion.* □

Next, we define the notion of neighbouring paths.

Definition 3.1.3. *Let* t *be an ordered tree, and let* t *contain node* o*. Let* $\alpha = (o, m_1, m_2, \ldots, m_r)$ *and* $\beta = (o, n_1, n_2, \ldots, n_s)$ *be two paths in* t*, for some* $r, s \geq 1$*, such that* o *is the parent of* m_1 *and* n_1*, where*

1. m_1 *is the direct left sibling of* n_1*;*
2. m_i *is the rightmost child of* m_{i-1}*, and* n_j *is the leftmost child of* n_{j-1}*,* $2 \leq i \leq r$*,* $2 \leq j \leq s$*.*

Then, α *and* β *are two* neighbouring paths *in* t*,* α *is the* left neighbouring path *to* β*, and* β *is the* right neighbouring path *to* α*.* □

Let us demonstrate the defined notions by the following example.

Example 3.1.1. Consider a graph $t = (V, E)$, where

$$\begin{aligned} V = \{ \quad & a, b, c, d, e, f, g, h, i, j, k, l, m, n, o, p, q, r, s \quad \}, \\ E = \{ \quad & (a, b), (a, c), (b, d), (b, e), (b, f), (c, g), \\ & (d, h), (e, i), (e, j), (f, k), (g, l), (g, m), (g, n), \\ & (i, o), (j, p), (l, q), (p, r), (q, s) \quad \}. \end{aligned}$$

Since in-d$(a) = 0$, in-d$(x) = 1$, and there exists a path from a to x, for any $x \in \{b, c, d, e, f, g, h, i, j, k, l, m, n, o, p, q, r, s\}$, t is a tree with the root node $\hat{r} = a$. The root a is the only node without any parent. It has two children b and c; b is a sibling of c and c is a sibling of b. Assume t is left-to-right ordered according to the illustration in Fig. 3.1.1.

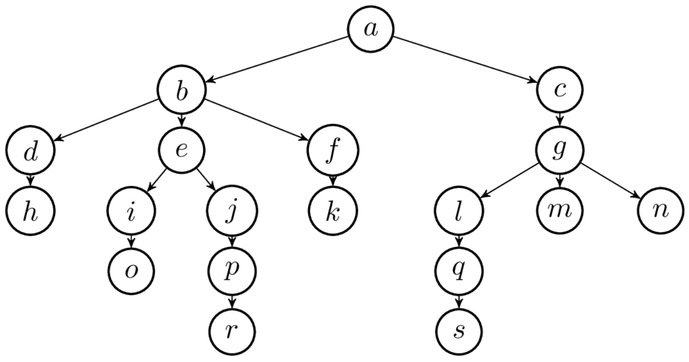


Figure 3.1.1: Labelled ordered tree t

Then, the leftmost child of b is d, while the rightmost is f. The node d is a left sibling of f, however, it is not the direct left sibling, which is e. The node f is the parent of k, but k has no child, so it is a leaf node. $horksmn =$ frontier(t).

Consider the node e. The nodes a and b are predecessors of e, while i, j, o, p, and r are e's descendants. The nodes c or d are not in predecessor relation with e, since they are neither predecessors of e, nor descendants of e. The node a is the lowest common predecessor of c and d, b is the lowest common predecessor of h and j, and a is the lowest common predecessor of a and p.

The sequence of nodes $bejpr$ is a path in t. The path bfk is neighbouring to $bejpr$; unlike $abfk$, eio, or bdh. □

3.2 Derivation Trees

We represent a generative process of a derivation of some grammar by a sequence of sentence forms together with applied rules. However, to more appropriately denote its hierarchical structure of rewritten symbols from the start symbol to terminal symbols we often use so-called derivation trees described in this section.

Definition 3.2.1. *Let t be a labelled ordered tree. A* left-bracketed representation *of t denoted by* lb-rep(t) *can be obtained by applying the following recursive rules:*

1. *If t has a root labelled $\hat{r}$ with subtrees $t_1, \dots, t_k$ ordered in this way, then*

$$\text{lb-rep}(t) = \hat{r}\langle \text{lb-rep}(t_1), \dots, \text{lb-rep}(t_k)\rangle.$$

   We sometimes omit separating commas if there is no risk of confusion.

2. *If t has a root labelled $\hat{r}$ with no direct descendants, then* lb-rep$(t) = \hat{r}$. □

Example 3.2.1. Consider labelled ordered tree t from Example 3.1.1. The left-bracketed representation of t is as follows.

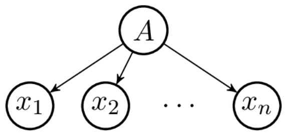
$$a\langle b\langle d\langle h\rangle e\langle i\langle o\rangle j\langle p\langle r\rangle\rangle\rangle f\langle k\rangle\rangle c\langle g\langle l\langle q\langle s\rangle\rangle mn\rangle\rangle\rangle$$

□

Based on the left-bracketed representation of a tree, we construct derivation trees for general grammars by the following procedure. Together with one-dimensional textual representation of the derivation tree we present graphical representation which is often more readable.

Definition 3.2.2. *Let $G = (V, T, P, S)$ be a GG (in the binary form).*

1. *For $p\colon A \to x \in P$, $A\langle x\rangle$ is the* rule tree *that represents p; assume $x = x_1x_2 \dots x_n$, where $x_i \in V$, for $1 \leq i \leq n$ ($n = 2$ for a GG in the binary form).*



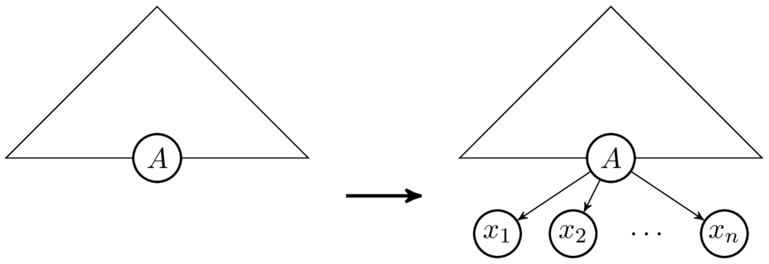
2. *The* derivation trees *representing derivations in G are defined recursively as follows:*

   (a) One-node tree with a node labelled X is the derivation tree corresponding to $X \Rightarrow^0 X$ in G, where $X \in V$. If $X = \varepsilon$, we refer to the node labelled X as ε-node *(*ε-leaf*); otherwise, we call it* non-ε-node *(*non-ε-leaf*).*

   (b) Let d be the derivation tree representing $X \Rightarrow^*_G uAv$ $[\varrho]$ with frontier$(d) = uAv$*, and let $p\colon A \to x \in P$. The derivation tree that represents*

$$X \Rightarrow^*_G uAv\ [\varrho] \Rightarrow_G uxv\ [p]$$

   is obtained by replacing the ith non-ε-leaf in d labelled A, with rule tree corresponding to p, $A\langle x\rangle$, where $i = |uA|$; assume $x = x_1x_2 \dots x_n$, where $x_i \in V$, for $1 \leq i \leq n$.

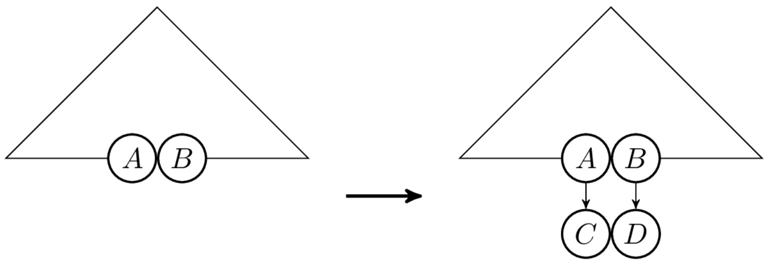


   (c) Let d be the derivation tree representing $X \Rightarrow^* uABv\,[\varrho]$ with frontier$(d) =$ *$uABv$, and let $p\colon AB \to CD \in P$. The derivation tree that represents*

$$X \Rightarrow^* uABv\ [\varrho] \Rightarrow uCDv\ [p]$$

   is obtained by replacing the ith and $(i+1)$th non-ε-leaf in d labelled A and B with $A\langle C\rangle$ and $B\langle D\rangle$, respectively, where $i = |uA|$.

3. *A* derivation tree *in G is any tree t for which there is a derivation represented by t (see 2 in this definition).* □



Note that if G is context-free, its derivation tree is obtained omitting 2c, since G has no non-context-free rules. Additionally, notice that a node labelled by ε is always a leaf-node—it denotes an erasion of a nonterminal.

After replacement in 2c, the nodes A and B are the parents of the new leaves C and D, respectively, and we say that A and B are *context-dependent*, alternatively speaking, we say that there is a context dependency between A and B. In a derivation tree, two nodes are *context-independent* if they are not context-dependent. If a node is labelled by terminal, it is called a *terminal node*; otherwise, it is a *nonterminal node*. If a node has more than one nonterminal child, it is called a *branching node*; otherwise, it is a *non-branching node*.

Definition 3.2.3. *Let $G = (V, T, P, S)$ be a GG. Then, for any $p\colon A \to x \in P$, ${}_G\triangle(p)$ denotes the rule tree corresponding to p. For any $A \Rightarrow^* x\,[\varrho]$ in G, where $A \in N$, $x \in V^*$, and $\varrho \in P^*$, ${}_G\triangle(A \Rightarrow^* x\,[\varrho])$ denotes the derivation tree corresponding to $A \Rightarrow^* x\,[\varrho]$. Just like we often write $A \Rightarrow^* x$ instead of $A \Rightarrow^* x\,[\varrho]$, we sometimes simplify ${}_G\triangle(A \Rightarrow^* x\,[\varrho])$ to ${}_G\triangle(A \Rightarrow^* x)$ in what follows if there is no danger of confusion. Let ${}_G\blacktriangle$ denotes the set of all derivation trees in G. Finally, by ${}_G\triangle_x \in {}_G\blacktriangle$, we mean any derivation tree whose frontier is x, where $x \in \mathscr{F}(G)$.* $\square$

The previous tree-related notions also apply for regular-controlled grammars, scattered context grammars and cooperating distributed grammar systems, as explained later, which this work also deals with, however, with one exception. For SCGs with their parallel rules Definition 3.2.2 is not satisfactory. Therefore, we present the following slightly modified definition of derivation trees of SCGs.

Definition 3.2.4. *Let $G = (V, T, P, S)$ be an SCG in the binary form.*

1. *For $p\colon (A) \to (x) \in P$, $A\langle x\rangle$ is the* rule tree *that represents p. For $p\colon (A, B) \to (x, y) \in P$, $A\langle x\rangle$ and $B\langle y\rangle$ are the rule trees that represent p. Graphical representation follows 1 of Definition 3.2.2.*

2. *The* derivation trees *representing derivations in G are defined recursively as follows:*

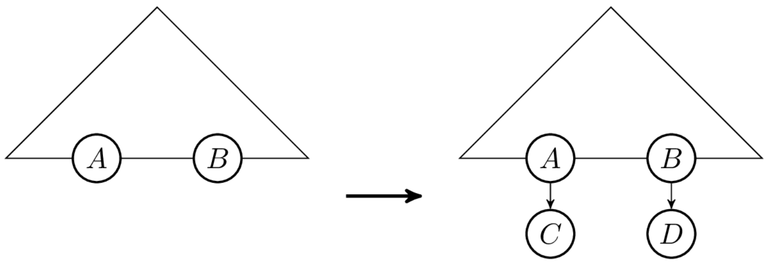
   (a) Follows 2a of Definition 3.2.2.

   (b) Follows 2b of Definition 3.2.2.

   (c) Let d be the derivation tree representing $X \Rightarrow^*_G uAvBw\ [\varrho]$ with frontier$(d) = uAvBw$*, and let $p\colon (A, B) \to (C, D) \in P$. The derivation tree that represents*

$$X \Rightarrow^*_G uAvBw\ [\varrho] \Rightarrow_G uCvDw\ [p]$$

   is obtained by replacing the ith and jth non-ε-leaf in d labelled A and B with $A\langle C\rangle$ and $B\langle D\rangle$, respectively, where $i = |uA|$ and $j = |uAvB|$.



3. *A* derivation tree *in G is any tree t for which there is a derivation represented by t (see 2 in this definition).* □

Remark that after replacement in 2c the nodes A and B are context-dependent similarly like in the case of GGs.

Example 3.2.2. Consider a grammar G from example Example 2.2.3. For the derivation $S \Rightarrow_G^* aabbcc$, the derivation tree ${}_G\triangle(S \Rightarrow_G^* aabbcc)$ is as follows.

$$S\langle A\langle \bar{A}\langle \bar{a}\langle a\rangle A\langle \bar{A}\langle \bar{a}\langle a\rangle A\langle \varepsilon\rangle\rangle\rangle\rangle\rangle B\langle \bar{B}\langle \bar{b}\langle b\rangle C\langle B\langle \bar{B}\langle \bar{b}\langle b\rangle C\langle B\langle \varepsilon\rangle \bar{c}\langle c\rangle\rangle\rangle\rangle \bar{c}\langle c\rangle\rangle\rangle\rangle$$

More often, we use the graphical representation of a derivation tree, as it is in Figrure 3.2.1, since the bracketed representation is clearly readable only for very small derivations.

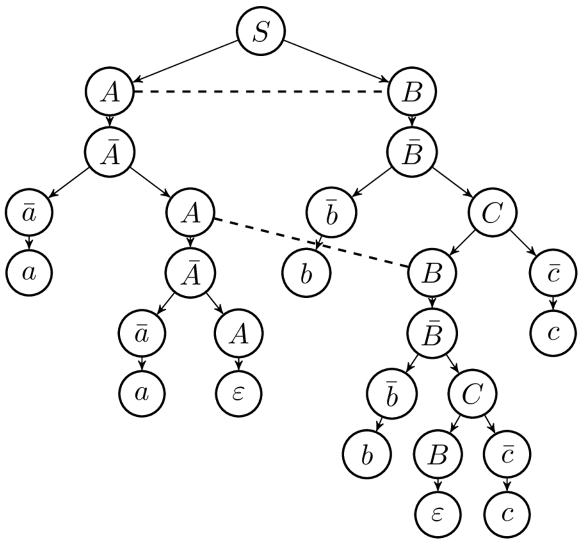


Figure 3.2.1: Graphical representation of ${}_G\triangle_{aabbcc}$

Let us remark that the dashed lines only denote the context-dependent nodes and are not a part of the derivation tree. □

In the following definition, we formalize mutual context dependencies between two neighbouring paths within a derivation tree.

Definition 3.2.5. *Let G be a general grammar and $t \in {}_G\blacktriangle$ be a derivation tree. Assume that $\alpha = (o, m_1, m_2, \ldots, m_r)$ and $\beta = (o, n_1, n_2, \ldots, n_s)$ are two neighbouring paths in t, where $r, s \geq 0$, α is the left neighbouring path to β, and m_r and n_s are leafs. Then, there is an l-tuple $\gamma = (g_1, g_2, \ldots, g_l)$ of nodes from α and l-tuple $\delta = (h_1, h_2, \ldots, h_l)$ of nodes from*

β, where $g_p < g_q$, for $1 \leq p < q \leq l$, $l < \min(r, s)$, and g_i and h_i are context-dependent, for $1 \leq i \leq l$. Let $\varrho = p_1 p_2 \ldots p_l$ be a string of non-context-free rules corresponding to context dependencies between γ and δ. We call ϱ the right context of α *and the* left context of β *or the* context of α and β. *Consider a node m_i, where $1 \leq i \leq r$, and two $(l-k+1)$-tuples of nodes $\sigma = (g_k, g_{k+1}, \ldots, g_l)$ and $\varphi = (h_k, h_{k+1}, \ldots, h_l)$, where k is a minimal integer such that $m_i < g_k$. Then, a string of non-context-free rules $\tau = p_k p_{k+1} \ldots p_l$ corresponding to context dependencies between σ and φ, for some $1 \leq k \leq l$, is called the* right descendant context of m_i. *Analogously, we define the notion of the* left descendant context *of a node n_j in β, for some $1 \leq j \leq s$.* □

We demonstrate the newly introduced notions by the following example.

Example 3.2.3. Let $G = (V, T, P, S)$ be a general grammar, where

$$V = \{S, S_a, S_b, X, X_a, X_b, Z_a, Z_b, A, 1, 2, 3, A_x, \overline{a}, a, B, B_x, \overline{b}, b\},$$

$T = \{a, b\}$, and P contains the following rules:

(1) $S \to S_a B_x$	(9) $X \to B X_a$	(17) $Z_b \to B$	(25) $A_x \to \overline{a}$
(2) $S \to S_b A_x$	(10) $X_a \to XA$	(18) $AB \to A_x B$	(26) $AA_x \to \overline{aa}$
(3) $S_a \to Z_a X$	(11) $X_b \to XB$	(19) $BA \to B_x A$	(27) $1A_x \to \overline{aa}$
(4) $S_b \to Z_b X$	(12) $Z_a A \to A Z_a$	(20) $BA_x \to B_x A_x$	(28) $2A_x \to \overline{aa}$
(5) $X \to XX$	(13) $Z_a B \to B Z_a$	(21) $AA \to \overline{a}1$	(29) $BB \to \overline{b} B_x$
(6) $X \to AB$	(14) $Z_b A \to A Z_b$	(22) $1A \to \overline{a}2$	(30) $B_x B \to \overline{b} B_x$
(7) $X \to BA$	(15) $Z_b B \to B Z_b$	(23) $2A \to \overline{a}3$	(31) $B_x B_x \to \overline{bb}$
(8) $X \to A X_b$	(16) $Z_a \to A$	(24) $3A_x \to \overline{aa}$	(32) $\overline{a} \to a$
			(33) $\overline{b} \to b$

At this point, let us make only an informal observation that $\mathrm{L}(G)$ is the language of all nonempty strings above T consisted of an equal number of as and bs, where every sequence of as is of a length between 1 and 5 and every sequence of bs is longer or equal 3. A rigorous proof comes later in Chapter 7.

The string $aabbba$ can be obtained by the following derivation:

$$\begin{array}{llllll}
S & \Rightarrow S_b A_x & [(2)] & & \Rightarrow Z_b X A_x & [(4)] \\
 & \Rightarrow Z_b A X_b A_x & [(8)] & & \Rightarrow Z_b A X B A_x & [(11)] \\
 & \Rightarrow Z_b A A B B A_x & [(6)] & & \Rightarrow A Z_b A B B A_x & [(14)] \\
 & \Rightarrow A A Z_b B B A_x & [(14)] & & \Rightarrow A A B B B A_x & [(17)] \\
 & \Rightarrow A A_x B B B A_x & [(18)] & & \Rightarrow A A_x B B B_x A_x & [(20)] \\
 & \Rightarrow \overline{aa} B B B_x A_x & [(26)] & & \Rightarrow \overline{aab} B_x B_x A_x & [(29)] \\
 & \Rightarrow \overline{aabbb} A_x & [(31)] & & \Rightarrow \overline{aabbba} & [(25)] \\
 & \Rightarrow a\overline{abbba} & [(32)] & & \Rightarrow aa\overline{bbba} & [(32)] \\
 & \Rightarrow aab\overline{bba} & [(33)] & & \Rightarrow aabb\overline{ba} & [(33)] \\
 & \Rightarrow aabbb\overline{a} & [(33)] & & \Rightarrow aabbba & [(32)]
\end{array}$$

A graph representing ${}_G\triangle(S \Rightarrow^* aabbba)$ is illustrated in Fig. 3.2.4.

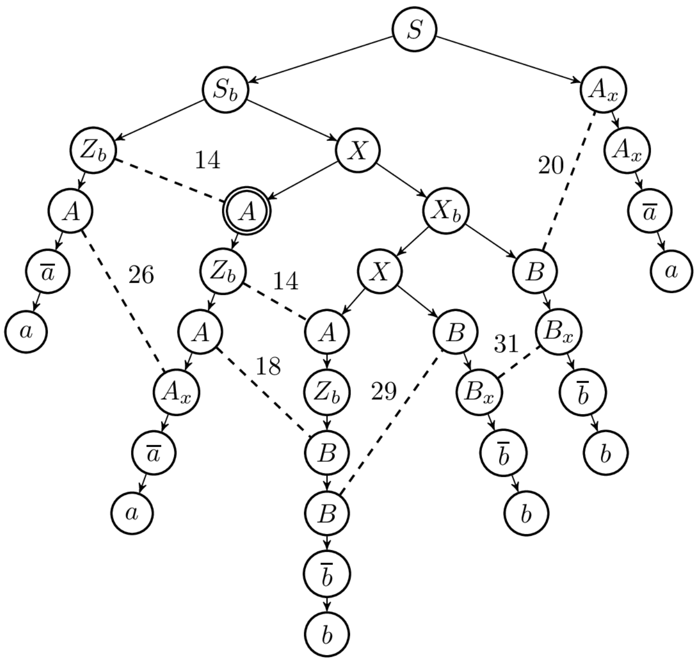


Figure 3.2.2: ${}_G\triangle_{aabbba}$

Let us note that dashed lines, numbers, and double circle contour only denote the context dependencies, applied non-context-free rules, and a specific node, respectively, and are not the part of the derivation tree.

Pairs of context-dependent nodes are linked with dashed lines, all the other nodes are context-independent. Since $aabbba = \text{frontier}({}_G\triangle_{aabbba})$, all the leafs are terminal nodes. Every other node is nonterminal node. For a pair of neighbouring paths $\alpha = S_b Z_b A\bar{a}a$ and $\beta = S_b X A Z_b A A_x \bar{a}a$, a string $\varrho = 14\ 26$ is their context, it is the left context of β and the right context of α. Consider the double circled node A. Then, $\tau = 26$ is the left descendant context of A and $\varphi = 14\ 18$ is the right descendant context of A. □

Two consecutive derivation steps may follow a common path with respect to the corresponding derivation tree or take place in different parts of it. Let us formalize this by the following definition.

Definition 3.2.6. *Let $G = (V, T, P, S)$ be a GG. Consider a derivation in G of length $n \geq 2$, $S \Rightarrow^n w$, for some $w \in V^*$. If the derivation is performed as*

$$S \Rightarrow^{n-2} uAv \Rightarrow uxByv \Rightarrow uxzyv,$$

where $uxzyv = w$, for some $u, v, w, x, y, z \in V^*$ and $A, B \in N$, we call the n-th step of the derivation path-preserving*—indeed, in a resulting derivation tree, the nodes corresponding to consecutively rewritten nonterminals A and B belong to the same path from $\hat{r}$. Otherwise, the derivation step is* path-changing*. By definition, the initial derivation step of any derivation is always path-preserving.* □

Example 3.2.4. Let $G = (\{S, A, B\}, \{a, b, c, d, e, f\}, P, S)$ be a CFG with

$$\begin{aligned} P = \{\ & 1\colon S \to aSb,\ \ 2\colon S \to AB, \\ & 3\colon A \to cAd,\ \ 4\colon A \to \varepsilon, \\ & 5\colon B \to eBf,\ 6\colon B \to \varepsilon\ \}. \end{aligned}$$

Obviously, $\mathrm{L}(G) = \{a^{n_1}c^{n_2}d^{n_2}e^{n_3}f^{n_3}b^{n_1} \mid n_1, n_2, n_3 \geq 0\}$ which is non-linear context-free language of index 2. Consider the following derivation.

$$S \Rightarrow_G aSb \Rightarrow_G aABb \Rightarrow_G acAdBb \Rightarrow_G acAdeBfb \Rightarrow_G acdeBfb \Rightarrow_G acdefb\,[123546]$$

A graph representing ${}_G\triangle(S \Rightarrow^*_G acdefb\,[123546])$ is illustrated in Fig. 3.2.3. Solid lines denote the edges of the tree, while the dashed lines denote the path of the derivation. If they overlap, the derivation step is path-preserving; otherwise, it is path-changing. By the definition, the initial derivation step is always path-preserving. Then, the graph demonstrates that there are precisely three path-changing derivation steps in $S \Rightarrow^*_G acdefb\,[123546]$. □

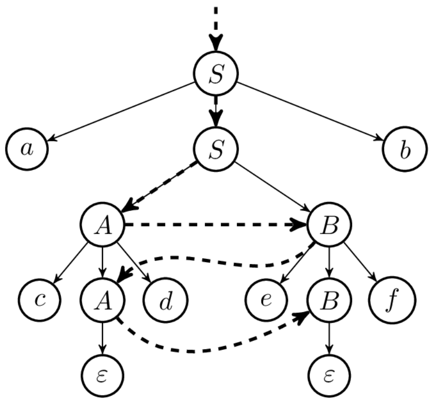


Figure 3.2.3: ${}_G\triangle(S \Rightarrow^* acdefb\,[123546])$

All the previous notation concerning CFGs (as a special case of GGs) and their derivation trees still applies for regular-controlled grammars, since their core grammars are CFGs. However, as the following example demonstrates, not every derivation of a core grammar is legal according to the control language.

Example 3.2.5. Let $H = (G, C)$ be an RCG with G from Example 3.2.4 and

$$C = \{1\}^*\{2\}\{3\}^*\{4\}\{5\}^*\{6\}$$

Obviously, G is in the binary form. Even though $\mathrm{L}(H) = \mathrm{L}(G)$, the derivation $S \Rightarrow^*_G acdefb\,[123546]$ from Example 3.2.4 is not legal in H, since $123546 \notin C$. However, $123456 \in C$ and, thus,

$$S \Rightarrow_H aSb \Rightarrow_H aABb \Rightarrow_H acAdBb \Rightarrow_H acdBb \Rightarrow_H acdeBfb \Rightarrow_H acdefb\,[123456]$$

in H. A graph representing ${}_G\triangle(S \Rightarrow^*_G acdefb\,[123456])$ is illustrated in Fig. 3.2.4.

Notice that there is only one path-changing derivation step despite the grammar is clearly of index 2. Indeed, a grammar with only linear rules is obviously of index 1 and no path-changing derivation steps occur in its derivations. For a grammar in the binary form the index as well as the minimal number of path-changing derivation steps increase

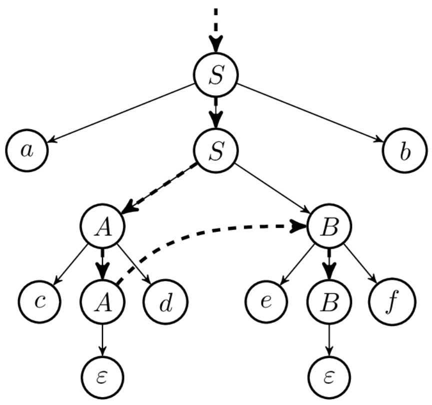


Figure 3.2.4: ${}_G\triangle(S \Rightarrow^* acdefb\,[123456])$

by one for every branching derivation step, because every branch must be ones terminated in a successful derivation. Therefore, if there is a constant k limiting the number of path-changes, the index of a grammar is at most $k + 1$ (for proof see Section 6.2). □

Next, we define a division of a tree into a set of connected subgraphs.

Definition 3.2.7. *Let $t = (V, E)$ be a tree. Define a k-*division *of a tree into a set of k connected subgraphs recursively as follows. Set t as 1-division of t itself. Let $t' = (V, E')$ be a k-division of t; t' is a set of k connected but mutually disconnected subgraphs. Construct $t'' = (V, E'')$ setting $E'' = E' - (u, v)$, where $(u, v) \in E'$. Then, t'' is $(k + 1)$-division of t. A* division *of a tree $t = (V, E)$ is any $t' = (V, E')$, where $E' \subseteq E$.* □

Let us illustrate k-division by the following example.

Example 3.2.6. Consider a tree $t = (V, E)$ from Example 3.1.1. A 4-division $t' = (V, E')$ of t, where $E' = E - \{(a, b), (b, f), (g, n)\}$,

$$\begin{array}{rll} V & = \{ & a, b, c, d, e, f, g, h, i, j, k, l, m, n, o, p, q, r, s \quad \}, \\ E' & = \{ & (a, c), (b, d), (b, e), (c, g), \\ & & (d, h), (e, i), (e, j), (f, k), (g, l), (g, m), \\ & & (i, o), (j, p), (l, q), (p, r), (q, s) \qquad\qquad \}, \end{array}$$

in fact represents a set of 4 separated connected subgraphs

$$\begin{array}{l} t_1 = (\{a, c, g, l, m, q, s\}, \{(a, c), (c, g), (g, l), (g, m), (l, q), (q, s)\}), \\ t_2 = (\{b, d, e, h, i, j, o, p, r\}, \{(b, d), (b, e), (d, h), (e, i), (e, j), (i, o), (j, p), (p, r)\}), \\ t_3 = (\{f, k\}, \{(f, k)\}), \\ t_4 = (\{n\}, \{\}). \end{array}$$

Figure 3.2.5 illustrates t'. □

Let us turn our attention to cooperating distributed grammar systems. Just point out that all the previous tree-related notion concerning CFGs as a special case of GGs can be easily generalized for CDGSs since their components are CFGs. Detailed definitions are, thus, left to the reader.

Definition 3.2.8. *Let $t = (V, E)$ be a tree and $(U, V - U)$ be a cut, $U \subseteq V$. $(U, V - U)$ is a* cut by layer *if graph generated by U, G_U, is a tree and $\hat{r} \in U$.* □

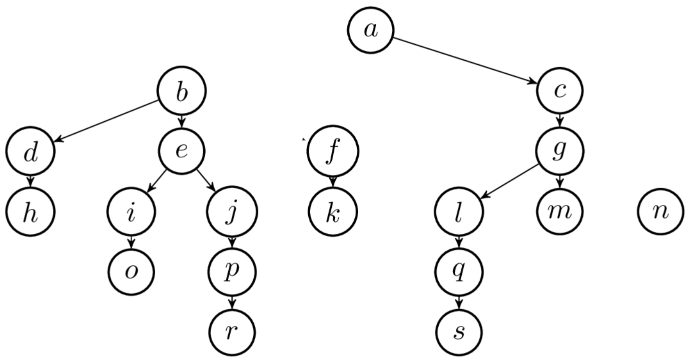


Figure 3.2.5: Tree division t'

A CDGS generates a sentence by possibly using several different components. Then, informally speaking, certain parts of a derivation tree correspond to the certain components of CDGS. The following definition describes this formally.

Definition 3.2.9. *Let n be a positive integer. Let $G = (V, T, S, P_1, \dots, P_n)$ be a CDGS of degree n, ψ be a derivation mode, and there is a derivation of a length $m \geq 2$*

$$S \Rightarrow_{i_1}^{\psi} w_1 \Rightarrow_{i_2}^{\psi} w_2 \Rightarrow_{i_3}^{\psi} \cdots \Rightarrow_{i_k}^{\psi} w_k \Rightarrow_{i_{k+1}}^{\psi} w_{k+1} \Rightarrow_{i_{k+2}}^{\psi} \cdots \Rightarrow_{i_m}^{\psi} w_m,$$

where $w_j \in V^*$, $1 \leq i_j \leq n$, $1 \leq j \leq m$, $1 \leq k \leq m-1$. If $i_k \neq i_{k+1}$, we call i_{k+1}th derivation step a component change*. Let the graph*

$$\mathcal{G} = (V, E) = {}_G\triangle(S \Rightarrow_{i_1}^{\psi} \cdots \Rightarrow_{i_m}^{\psi} w_m)$$

be a derivation tree corresponding to the previous derivation. A cut $(U, V-U)$ is a component-change cut *(${}_{cc}$cut for short) if for the graph generated by U, $\mathcal{G}_U$, $\mathcal{G}_U = (U, E_U) = {}_G\triangle(S \Rightarrow_{i_1}^{\psi} \cdots \Rightarrow_{i_k}^{\psi} w_k)$.* □

Notice that every component-change cut is also a cut by layer.

Example 3.2.7. Let $G = (\{S, A\}, \{a\}, P_1, P_2, P_3, S)$ be a CDGS with

$$P_1 = \{1\colon S \to AA\}, P_2 = \{2\colon A \to S\}, P_3 = \{3\colon S \to a\}.$$

and consider the derivation mode t. Then, obviously, $L(G^t) = \{a^{2^k} \mid k \geq 0\}$ which is a well-known non-context-free context-sensitive language. Consider the following derivation.

$$d = S \Rightarrow_1^t AA \Rightarrow_2^t SA \Rightarrow_2^t SS \Rightarrow_3^t aS \Rightarrow_3^t aa$$

A graph representing ${}_G\triangle(d)$ is illustrated in Fig. 3.2.6. Moreover, it represents various cuts. Remark that the numbers represent nodes themselves while the symbols from $\{S, A, a\}$ represent node labels corresponding to the derivation.

First, the dotted line denotes the cut ($\{2,3\}$,$\{1,4,5,6,7\}$) which is neither cut by layer, nor component-change cut. Second, the dashed line denotes the cut ($\{1,2,3,5\}$,$\{4,6,7\}$) which is a cut by layer, since a graph generated by $\{1,2,3,5\}$ is a tree and contains a root of ${}_G\triangle(d)$, however, not a component-change cut, since $\text{alph}(SA) \cup \text{dom}(P_2) \neq \emptyset$ and, thus, there cannot follow a component change in t derivation mode. Finally, the two solid lines represent the cuts ($\{1,2,5\}$,$\{3,4,6,7\}$) and ($\{1,2,3,5,6\}$,$\{4,7\}$) which are the only possible component-change cuts of ${}_G\triangle(d)$. □

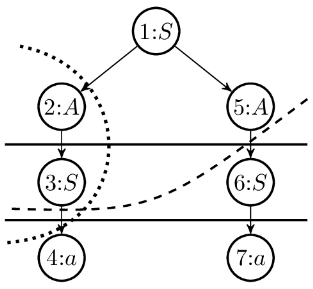


Figure 3.2.6: ${}_G\triangle(d)$

Finally, let us point out that neither representation covers the full information about the whole derivation process alone and we usually combine them. Indeed, since there may be numerous possibilities how to apply a single rule, having only the sequence of applied rules need not to give us the exact knowledge of the resulting sentence. Conversely, having only the sequence of intermediate sentence forms may give us insufficient information on of the applied rules in the case there are more of them acting similarly. Both of these representations give us no direct inside into the hierarchical tree structure of the rewritten symbols; of course, we may reconstruct this from the knowledge of the applied rules or intermediate sentence forms. On the other hand, having only a derivation tree for the resulting terminal sentence does not give us any information on the order of applied rules during the derivation. Therefore, we usually use multiple different representations of the derivation at once to get that certain advantage of each of them.

Chapter 4

Proof Techniques in Formal Language Theory

In this two-section chapter for each of the main formal language families we describe process of obtaining positive or negative proof of membership of a language in a the family.

4.1 Positive Proofs

Theory of formal languages always intensively struggled for a certain language to effectively determine to which family of languages it precisely belongs to. Nevertheless, it is a straightforward task only for very simple languages and, thus, a challenging area for researchers to investigate. There exist some well-known procedure patterns following which we can obtain a positive proof that some language is for example regular or context-sensitive, however, let us point out that this is always a creative process; or we can say that there exists no algorithm for proving of language-family membership.

Obviously, we can always perform this by constructing a grammar or an automaton corresponding to a certain language family and showing that it describes the language in question. However, it is usually not as easy as it may seem to be, so we often tend to determine some sufficient conditions under which, e.g., a more powerful type of grammars generates a certain subfamily of languages, since it is easier to describe the language with more powerful tool. Then, we have to show that the grammar satisfies those conditions which immediately results in the proof of membership of the language in the subfamily. There are, sadly, only a few language families among the most well-knowns, which we present in Chapter 5, known for having such sufficient conditions for a language to belong to them—with workspace condition for context-sensitive languages in the front which is also a subject of this section.

In the present section, we first briefly discuss the positive proofs of regularity, since necessary but also sufficient conditions for a language to be regular are introduced in Section 4.2 which gives the detailed explanation of several so-called pumping lemmas for regular languages. The positive proofs of context-freeness are the main matter of this work, however, in this section we present only the former results. The reader may find the new ones in Chapter 6 with the detailed explanation together with practical examples in Chapter 7. The main focus of the present section is directed to the detailed explanation of workspace conditions for context-sensitive languages (the proof was taken from [48]). Finally, we discuss conditions for a languages to be recursively enumerable.

How to Prove Regularity

The most straightforward approach to proving regularity of a language L is constructing a regular grammar G or a finite automaton M and showing that $\mathrm{L}(G) = L$ or $\mathrm{L}(M) = L$, respectively. Since the family of regular languages is in fact a smallest and, thus, the less complicated abstract family of languages in Chomsky hierarchy described later in Chapter 5, it is often a viable option.

On the other hand, observe Definition 2.1.2 of regular language. Based on this we can simply prove a regularity a of a language without constructing any grammar or automaton by showing that we can obtain the language by a finite number of basic string (or language) operations. We demonstrate this by the following example.

Example 4.1.1. Consider a language L of all strings of odd length above alphabet $\Sigma = \{a, b\}$. Let $L_{\text{pair}} = \{aa, ab, ba, bb\}$ be the language of all pairs of symbols above Σ. Then,

$$\Sigma L_{\text{pair}}^*$$

denote the language L and, thus, $L \in \mathbf{REG}$. □

Sometimes we need to show that some more powerful, usually context-free, grammar generates in fact a regular language. Then, we can obtain a proof of this by showing that the grammar satisfies some sufficient conditions for a language to be regular. We skip explanation of this possible approach for showing regularity here and postpone it to Section 4.2, where we present two so-called pumping lemmas which introduce necessary and also sufficient conditions for regularity. They are mostly used to disprove regularity, however, they may also serve for the opposite purpose. Nevertheless, let us note that the problem of regularity is generally undecidable for context-free languages (see [22]) and, thus, always a matter of a creative proof process.

How to Prove Context-Freeness

Effective proving of the context-freeness of a language is the main subject of this work. We can, obviously, construct a context-free grammar and show that it generates the language in question as well as we can make this kind of construction prove in the case of showing regularity. However, it indisputably requires a unique creative proof process. There exists well-known pumping lemma for context-free languages (see Section 4.2) which is widely used to demonstrate that certain languages are beyond the power of context-free grammars, but we lack any proof pattern analogical to this at least generally describing some necessary steps to obtain a positive proof of context-freeness which we could in some sense mechanically follow.

Ideally, we would take some more powerful grammar, for example general, by which we can more easily describe the given language. Then, by showing that it obeys a certain restrictions prove that the generated language is in fact context-free. Unfortunately, there are no such restrictions representing a sufficient conditions for context-freeness known so far. However, in Chapter 6 we introduce such restrictions for not only general grammars, but also parallel and regulated grammars and grammar systems, represented by scattered context and regular-controlled grammars and cooperating distributed grammar systems, respectively. Moreover, in Chapter 7 we give some practical examples of showing context-freeness to demonstrate the application perspectives of the newly acquired results.

How to Prove Context-Sensitivity

Observe that in a nonempty sentential form in a context-sensitive grammar the length of the consecutive sentential forms is increasing monotonically. This means that in a derivation of a terminal string w all the sentential forms have the length less than or equal to $|w|$. Now assume that a language L is generated by a general grammar such that there is a nonnegative integer k with the property that for each sentence w, $w \in \mathrm{L}(G)$, there exists a derivation of w in G such that the workspace does not exceed $k|w|$, that is there is a derivation $S \Rightarrow_G^* w$ such that all sentential forms from the derivation have the length less than or equal to $k|w|$.

If G has the above property, then $\mathrm{L}(G)$ is a context-sensitive language. Let us define the notion formally as follows.

Definition 4.1.1. *Let $G = (V, T, P, S)$ be a general grammar and consider a derivation d of a string w according to grammar G,*

$$d \colon S = w_0 \Rightarrow_G w_1 \Rightarrow_G \cdots \Rightarrow_G w_n = w.$$

for some $n \geq 0$. The workspace *of w by the derivation d is*

$$\mathrm{WS}_G(w, d) = \max\{|w_i| \mid 0 \leq i \leq n\}.$$

The workspace of w is

$$\mathrm{WS}_G(w) = \min\{\mathrm{WS}_G(w, d) \mid d \textit{ is a derivation of } w\}.$$

Observe that $\mathrm{WS}_G(w) \geq |w|$ for all G and w. □

The following theorem, due to Jones (see [25]), is a powerful tool in showing languages to be context-sensitive.

Theorem 4.1.1 (Workspace Theorem). *If G is a general grammar and if there is a nonnegative integer k such that*

$$\mathrm{WS}_G(w) \leq k|w|$$

for all nonempty sentences $w \in \mathrm{L}(G)$, then $\mathrm{L}(G)$ is a context-sensitive language. □

Proof. Basic Idea. Let $G = (V, T, P, S)$ be a general grammar with $\mathrm{L}(G) = L$ satisfying workspace theorem for some nonnegative integer k. Since the workspace of general grammar never extends $k|w|$, for any sentence $w \in L$, we can simulate general grammar G by some context-sensitive grammar (or monotone general grammar) $G' = (V', T, P', S')$ as follows—we give just a gist of the construction. Define nonterminal symbols of V' as a compositions of k (or less) symbols $\langle A_1|A_2|\cdots|A_k\rangle$, $A_i \in V$, $1 \leq i \leq k$, and define the rules of P' to work above these compositions in the way the rules of P do with the separate symbols. Then, obviously, we can simulate the generative process of G above workspace of the length $k|w|$ with only $|w|$ long workspace. □

We refer the reader to [50] for the detailed proof of the above theorem. An immediate consequence is the following corollary.

Corollary 4.1.1. *Let L be a recursively enumerable language that is not a context-sensitive language. Then, for every nonnegative integer k and for every general grammar G generating L, there is a sentence $w \in L$ such that $\mathrm{WS}_G(w) \geq k|w|$.* □

By the application of this corollary we can in fact prove that some languages are beyond the power of context-sensitive grammars and, thus, non-context-sensitive, as shown in Section 4.2 which discusses negative proofs. In this chapter we instead demonstrate how to apply Workspace Theorem to prove that some language is context-sensitive.

Example 4.1.2. Consider a language above the unary alphabet

$$L = \{w \in \{a\}^* \mid 3|w| = 2^n + 1 \text{ for some } n \geq 1\}.$$

Since the language of all unary strings with length of a power of 2 is the well-known context-sensitive language (see [43]), we can guess that L is also context-sensitive. However, it may be a bit tricky to show this by constructing a context-sensitive or monotone general grammar, since it has to work in only a one third of usual workspace. Nevertheless, we may perform this by constructing a general grammar and showing that it satisfies workspace conditions for context-sensitive languages.

Construct a general grammar $G = (\{A, B, C, D, E, F, X, a\}, \{a\}, P, S)$ with

$$\begin{array}{rrl} P = \{ & 1: & S \to ACXB, \\ & 2: & CX \to XXC, \\ & 3: & CB \to DB, \\ & 4: & CB \to E, \\ & 5: & XD \to DX, \\ & 6: & AD \to AC, \\ & 7: & XE \to EX, \\ & 8: & AE \to FX, \\ & 9: & FXXX \to aF, \\ & 10: & F \to \varepsilon \qquad\qquad \} \end{array}$$

and let us investigate the language $\mathrm{L}(G)$. First, after using the initial rule 1, G applies the following rules in a loop. By rule 2, C moves from left to right while doubling the number of Xs. A serves as the left border, where D is rewritten back to C by rule 6, while B represents the right border, where C is rewritten to D by rule 3. By rule 5, D is moved again from right to left. As a result

$$S \Rightarrow_G^* AX^kCB$$

where $k = 2^n$ for some $n \geq 1$. Eventually, CB is rewritten to E by rule 4 and E is moved to the right by rule 7. Then, by rule 8

$$AX^kCB \Rightarrow_G AX^kE \Rightarrow_G^k AEX^k \Rightarrow_G FX^{k+1}.$$

Finally, by rule 9 every three occurrences of X from left to right are rewritten to a single a and the derivation finishes by rule 10. However, to successfully rewrite all Xs to as $k+1$ needs to be a multiple of 3.

Clearly, $\mathrm{L}(G) = L$. Let us analyze the workspace of G. Initial rule extends the workspace from 1 to 4 with a single X. Then, the number of Xs is extended to 2^n for some $n \geq 1$, generating the string $AX^{2^n}CB$ with workspace of the length $2^n + 3$. In the rest of the derivation G applies the rules which preserves or shorten the sentential form. Therefore, $2^n + 3$, for some $n \geq 1$, is the length of the longest sentential form within any derivation and thus $\mathrm{WS}_G(w) = 2^n + 3$ for every $w \in \mathrm{L}(G)$, where $|w| = (2^n + 1)/3$. Therefore,

$$\frac{\mathrm{WS}_G(w)}{|w|} = \frac{2^n+3}{\frac{2^n+1}{3}} = \frac{2^n+1+2}{\frac{2^n+1}{3}} = \frac{(2^n+1+2)*3}{2^n+1} = \frac{(2^n+1)*3+6}{2^n+1} = 3 + \frac{6}{2^n+1}$$

The obtained expression is dependent on the length of the sentence, however, since the value is decreasing, we can easily determine that its maximum is obtained for the minimal possible length which is $n = 1$. Then,

$$3 + \frac{6}{2^1 + 1} = 5 = m$$

where m is the Workspace Theorem constant which proves that L is, indeed, a context-sensitive language. □

How to Prove Recursive Enumerability

The family of recursively enumerable languages in fact represents the class of all existing algorithmically solvable problems. We usually characterize it by introducing some so-called computationally complete formal models as Turing machines and recursive functions (see [45, 54]) or general grammars which we present in this work. All of them were shown to be equivalent in terms of descriptive power. Nevertheless, any such formal characterization of **RE** indisputably follows from the fundamental Turing-Church thesis.

Turing-Church Thesis. *Let L be a language. Then, $L \in$ **RE** if and only if there is a procedure that defines L by listing all its strings.*

Observe that Turing-Church thesis is indeed a thesis, not a theorem because it cannot be proved, since it is based on an intuitive notion of procedure. Originally, Turing-Church thesis have been stated in terms of Turing machines in [54], however, as general grammars and Turing machines are equivalent (see [35]), we consider a general grammar as a model of this intuitive procedure which is obviously perfectly correct and legal from a mathematical viewpoint. Alongside with this, there exists also well-known characterization of recursively enumerable languages by context-free languages.

Theorem 4.1.2 (see [18]). *For every recursively enumerable language K, there exist two context-free languages, L_1 and L_2, and a homomorphism h such that*

$$K = h\big(L_1 \cap L_2\big)$$

Determination of whether a language is recursively enumerable or not is rarely a straightforward process, since non-context-sensitive recursively enumerable languages represent enormously complicated types of problems. Proving that a certain language is recursively enumerable is often a question of whether there exists an algorithm to solve the task which the language represents or not and, thus, rather a subject of decidability. Unlike workspace conditions for context-sensitive languages, there exists no general pattern for showing a language to be recursively enumerable.

Moreover, there are languages beyond the family of recursively enumerable languages, as shown in Chapter 5, which, thus, cannot be recognized by any existing procedure. There exist several formally introduced procedures how to show that some language is not recursively enumerable and, therefore, beyond the power of general grammars; usually reduction to Post correspondence (see [46]) or other well-known unsolvable problem.

4.2 Negative Proofs

There are several ways to classify a language into a specific family of languages; for example, this can be done by demonstrating that the language is generated by a corresponding grammar or accepted by a corresponding automaton. Nevertheless, we sometimes cannot easily introduce a direct prove. However, by disclaiming a membership of a language in a certain family of languages we can indirectly prove that the language belongs to some superfamily of languages. More specifically, first we need to determine precise properties that every language belonging to some family of languages must fulfill. Then, by showing that a certain language does not meet these properties we also prove that it does not belong to this language family.

To prove that a language is for example not regular or context-free, the most commonly used tools are the *pumping properties* of languages, which are usually stated as *pumping lemmas*. The term pumping intuitively describes the property that any sufficiently long sentence of the language has a nonempty substring that can be so-called pumped. This means that if the substring is replaced by an arbitrary number of copies of the same substring, the resulting sentence is still in the language. Throughout this section, we introduce technique of negative proofs by several pumping lemmas. Theoretical subjects are always followed by examples of how to utilize it practice.

First, we show that there are necessary conditions for a language to be regular (we follow Section 4.1 in [48]). Moreover, two of the introduced pumping lemmas specify not only necessary but also sufficient conditions for a language to be regular. Therefore, even though they are mainly used to disprove regularity of a language, we may actually use them to prove that a certain language is regular. Next, we turn to context-free languages. Based on [38], we introduce another pumping lemma which specifies a necessary conditions for a language to be context-free and, thus, by which we can disprove context-freeness as well. However, as stated in [48], we lack knowledge of a sufficient conditions for a language to be context-free—which is, indeed, a subject of this work. Then, as a special case of context-freeness, we demonstrate how to disprove linearity by introducing a pumping lemma for linear languages. Finally, we utilize Work Space theorem for context-sensitive languages introduced in the previous section in a proof of non-context-sensitivity.

How to Disprove Regularity

There are many versions of pumping lemmas for regular languages. The most commonly used version is necessary but not sufficient condition for regularity; every regular language satisfies these conditions, but those conditions do not necessarily imply regularity. The first necessary and sufficient condition was introduced by Jaffe in [24]. Another necessary and sufficient pumping lemma, which is called block pumping, was established by Ehrenfeucht, Parikh, and Rozenberg in [15]. It is in contrast with context-freeness of languages, for which only some necessary pumping conditions are known, but no conditions are known to be also sufficient (see [48]).

In the following, we describe four pumping lemmas for regular languages; two necessary pumping lemmas and two necessary and sufficient pumping lemmas. We will give a proof for the first and the third, but omit the proofs for the second and the fourth. Examples will also be given to show these lemmas can be used to prove the non-regularity of a certain languages.

The first pumping lemma below was originally formulated in [3] and has appeared in many introductory books (see [8, 22, 23, 51, 57]).

Lemma 4.2.1. *Let L be a regular language over Σ. Then, there is a constant k, depending on L, such that for each $w \in L$ with $|w| \geq k$ there exist $x, y, z \in \Sigma^*$ such that $w = xyz$ and*

1. $|xy| \leq k$,
2. $|y| \geq 1$,
3. $xy^tz \in L$ *for all* $t \geq 0$.

Proof. Since $L \in$ **REG**, there exists a DFA $M = (Q,\ \Sigma,\ R,\ s,\ F)$ (see Definition 2.3.2), where $k = |Q|$ is the number of states of M and $\mathrm{L}(M) = L$. For a string $w = a_1a_2 \dots a_n \in L$, we denote the computation of M on w by the sequence of transitions

$$q_0a_1a_2 \dots a_n \vdash q_1a_2 \dots a_n \vdash \dots \vdash q_{n-1}a_n \vdash q_n$$

where $q_0, q_1, \dots, q_n \in Q$, $q_0 = s$, $q_n \in F$, and $q_ia_{i+1} \to q_{i+1} \in R$ for all i, $0 \leq i < n$.

If $n \geq k$, the above sequence has states q_i and q_j, $0 \leq i < j \leq n$, such that $q_i = q_j$. Then, for each $t \geq 0$, we have the transition sequence

$$\begin{aligned} q_0a_1 \dots \{a_{i+1} \dots a_j\}^t \dots a_n &\vdash^* q_i\{a_{i+1} \dots a_j\}^t \dots a_n \\ &\vdash^* q_i\{a_{i+1} \dots a_j\}^{t-1} \dots a_n \\ &\vdash^* q_i\{a_{i+1} \dots a_j\}^{t-2} \dots a_n \\ &\ \vdots \\ &\vdash^* q_i\{a_{i+1} \dots a_j\}^{2} \dots a_n \\ &\vdash^* q_i\{a_{i+1} \dots a_j\}^{1} \dots a_n \\ &\vdash^* q_i\{a_{i+1} \dots a_j\}^{0} \dots a_n \\ &\vdash^* q_n \end{aligned}$$

also in M. Let $x = a_1a_2 \dots a_i$, $y = a_{i+1} \dots a_j$, and $z = a_{j+1} \dots a_n$. Then, $xy^tz \in L$ for all $t \geq 0$, where $|xy| \leq k$ and $|y| \geq 1$. □

The lemma states that every regular language possesses the above pumping property. Therefore, any language that does not possess the property is not a regular language. When proving non-regularity, we usually proceed in the following way.

(1) Assume that L is regular.

(2) Select a string $w \in L$ whose length depends on the pumping-lemma constant k so that $|w| \geq k$ is necessarily true.

(3) For all possible decompositions of w into xyz satisfying the pumping-lemma conditions, find $t \geq 0$ such that $xy^tz \notin L$, which contradicts Lemma 4.2.1.

(4) The contradiction obtained in (3) means that the assumption in (1) is incorrect; therefore, L is not regular.

Let us demonstrate the proof of non-regularity by the following example.

Example 4.2.1. Consider a language $L = \{a^nb^n \mid i \geq 0\}$. Assume that L is regular and k is a pumping-lemma constant. Chose a sentence $w = a^kb^k$ which is obviously in L. Clearly, $|w| \geq k$. By the pumping lemma, $w = xyz$ for some $x, y, z \in \Sigma^*$ such that

1. $|xy| \leq k$,
2. $|y| \geq 1$, and
3. $xy^t z \in L$ for all $t \geq 0$.

By 1. and 2., we have $y = a^m$, for some $1 \leq m \leq k$. However, $xy^0z = xz = a^{k-m}b^k$ is not in L. Thus, 3. does not hold, and, therefore, L does not satisfy the pumping property of Lemma 4.2.1. □

The pumping lemma has used to show the non-regularity of many languages, e.g., the set of all binary numbers whose value is prime [22], the set of all palindromes over a finite alphabet [23], or the set of all strings of length i^2 for $i \geq 0$ [57].

However, not only regular languages but also some non-regular languages satisfy the pumping property of Lemma 4.2.1 as shown in the following example.

Example 4.2.2. Let $L \subseteq \Sigma^*$ be an arbitrary non-regular language and $L_\# = \#^+L$, where $\# \notin \Sigma$. Then, $L_\#$ satisfies the conditions of Lemma 4.2.1 with the constant k being 1. For any string $w \in \#^+L$, we can chose $x = \varepsilon$ and $y = \#$. However, $L_\#$ is not regular, which can be shown as follows. Let h be a morphism defined by $h(a) = a$ for each $a \in \Sigma$ and $h(\#) = \varepsilon$. Then, obviously, $L = h(L_\#)$. Assume that $L_\#$ is regular. Then, L is regular since regular languages are closed under morphism (see [50]). Nevertheless, this contradicts the assumption. Thus, $L_\#$ is not regular. □

Note that for each language $L \subseteq \Sigma^*$, we can construct a distinct language $L_\# \subseteq (\Sigma \cup \{\#\})^*$ that satisfies Lemma 4.2.1. Consequently, there are uncountably many non-regular languages that satisfy the pumping lemma.

Below, we give two more examples of non-regular languages that satisfy the pumping condition of Lemma 4.2.1, which are quite simple and interesting.

Example 4.2.3. Let $L \subseteq b^*$ be an arbitrary non-regular language. Then, the languages

1. $a^+L \cup b^*$ and
2. $aL \cup aa^+\{a, b\}^* \cup b^*$

are non-regular, but satisfy the pumping condition of Lemma 4.2.1. □

Next, we introduce the second pumping lemma for regular languages.

Lemma 4.2.2. *Let L be a regular language over Σ. Then, there is a constant k depending on L such that for all $u, v, w \in \Sigma^*$, if $|w| \geq k$, then there exist $x, y, z \in \Sigma^*$, $y \neq \varepsilon$ such that $w = xyz$ and for all $t \geq 0$ it holds that $uxy^tzv \in L$ iff $uwv \in L$*

Any language that satisfies the pumping condition of Lemma 4.2.2 satisfies also the pumping condition of Lemma 4.2.1. This follows by setting $u = \varepsilon$ and $|w| = k$ in the condition of Lemma 4.2.2. However, the converse is not true. We can show that there exist languages that satisfy the pumping condition of Lemma 4.2.1, but do not satisfy that of Lemma 4.2.2. For example, let $L = \{a^nb^n \mid n \geq 0\}$ and consider the language $L_\# = \#^+L$ as in Example 4.2.2. Clearly, $L_\#$ satisfies the pumping condition of Lemma 4.2.1. However, if we chose $u = \#$, $v = \varepsilon$, and $w = a^kb^k$ for Lemma 4.2.2, where k is the constant (corresponding to $L_\#$), it is clear that there do not exist x, y, z as required by the lemma.

Therefore, the set of languages that satisfy the pumping condition of Lemma 4.2.2 is a proper subset of the set of languages that satisfy the condition of Lemma 4.2.1. In other words, Lemma 4.2.2 can rule out more non-regular languages. In this sense, we say that Lemma 4.2.2 is a stronger pumping lemma for regular languages than Lemma 4.2.1.

Nevertheless, Lemma 4.2.2 still does not give a sufficient condition for regularity. We show in the following that there exist non-regular languages that satisfy the pumping condition of Lemma 4.2.2. In fact, the number of such languages is uncountable. A different proof was given in [15].

Example 4.2.4. Let L ba an arbitrary non-regular language over Σ and $\$ \notin \Sigma$. Define

$$L_{\$} = \{\$^+ a_1 \$^+ a_2 \$^+ \ldots \$^+ a_m \$^+ \mid a_1 a_2 \ldots a_m \in L, m \geq 0\}.$$

We can easily prove that $L_{\$}$ is non-regular. Let $\Sigma_{\$}$ denote $\Sigma \cup \{\$\}$. We now show that $L_{\$}$ satisfies the pumping condition of Lemma 4.2.2. Let $k = 3$ be the constant for the pumping lemma. To establish the nontrivial implication of the statement of the lemma, it suffices to show that for any $u, w, v \in \Sigma_{\$}^*$ with $uwv \in L$ and $|w| \geq 3$, there exist $x, y, z \in \Sigma_{\$}^*$ with $w = xyz$ and $y \neq \varepsilon$ such that $uxy^i zv \in L_{\$}$ for all $i \geq 0$. We can simply chose $y = \$$. □

The next pumping lemma, introduced by Jaffe [24], gives a necessary and sufficient condition for regularity. A detailed proof of the following lemma can be found also in [51].

Lemma 4.2.3. *A language $L \subseteq \Sigma^*$ is regular iff there is a constant $k \geq 1$ such that for all $w \in \Sigma^*$, if $|w| \geq k$ then there exist $x, y, z \in \Sigma^*$ such that $w = xyz$ and $y \neq \varepsilon$, and for all $i \geq 0$ and all $v \in \Sigma^*$, $wv \in L$ iff $xy^i zv \in L$.*

Proof. Only If. The only if part is relatively straightforward. Let M be a complete DFA that accepts L and k the number of states of M. For any string w of length $l \geq k$, e.g., $w = a_1 a_2 \ldots a_l$, let the state transition sequence of M on w be

$$q_0 a_1 a_2 \ldots a_l \vdash q_1 a_2 \ldots a_l \vdash \cdots \vdash q_{l-1} a_l \vdash q_l$$

where q_0 is the start state. Since there are at most k distinct states among $q_0, q_1, \ldots, q_l$ and $k < l + 1$, it follows that $q_i = q_j$ for some $0 \leq i < j \leq l$. This implies that the transition from q_i to q_j is a loop back to the same state. Let $x = a_1 \cdots a_i$, $y = a_{i+1} \cdots a_j$, and $z = a_{j+1} \cdots a_l$ ($x = \varepsilon$ if $i = 1$ and $z = \varepsilon$ if $j = l$). Then, for all $i \geq 0$,

$$q_0 xy^i z \vdash^* q_l,$$

so M is in the same state q_l after reading each string $xy^i z$, $i \geq 0$. Therefore, for all $i \geq 0$ and for all $v \in \Sigma$, $xy^i zv \in L$ iff $wv \in L$.

If. Let L be a language which satisfies the pumping condition of the lemma and k be the constant. We prove that L is regular by constructing a DFA M_L using the pumping property of L and, then, proving that $\mathrm{L}(M_L) = L$.

Construction. The DFA $M_L = (Q, \Sigma, R, s, F)$ is defined as follows. Each state in Q corresponds to a string w, in Σ^*, with length less than k, i.e.,

$$Q = \{q_w \mid w \in \Sigma^* \text{ and } |w| \leq k - 1\},$$

$s = q_\varepsilon$ and $F = \{q_w \in Q \mid w \in L\}$. The set of transition rules R is defined as

1. If $|w| < k-1$, then for each $a \in \Sigma$,

$$q_w a \rightarrow q_{wa}.$$

2. If $|w| = k-1$, then by the pumping property of L, for each $a \in \Sigma$, wa can be decomposed into xyz, $y \neq \varepsilon$, such that for all $v \in \Sigma^*$, $xyzv \in L$ iff $xzv \in L$. There may be a number of such decompositions. We chose the one such that xy is the shortest (and y is the shortest if there is a tie). Then, define

$$q_w a \rightarrow q_{xz}.$$

Now we show that the language accepted by M_L is exactly L. We prove this by induction on the length of a string $w \in \Sigma^*$.

Basis. Observe the construction of M_L. It is clear that for all words that $|w| < k$, $w \in \mathrm{L}(M_L)$ iff $w \in L$ by the definition of M_L.

Induction Hypothesis. Suppose that the lemma holds for all strings w shorter than some n, where $n \geq k$.

Induction Step. Consider a string $w \in \Sigma$ with $|w| = n$. Let $w = w_0 v$, where $|w_0| = k$. By the construction of M_L, we have

$$sw \vdash^* q \quad \text{and} \quad sxz \vdash^* q$$

where $q = q_{xz}$ for some $x, z \in \Sigma^*$ and $w_0 = xyz$, $y \in \Sigma^+$, and for any $v' \in \Sigma^*$, $w_0 v' \in L$ iff $xzv' \in L$. We replace the arbitrary v' by v, then we have that $w \in L$ iff $xzv \in L$. Since xz and w_0 reach the same state in M_L, xzv and $w = w_0 v$ will reach the same state, i.e., $w \in \mathrm{L}(M_L)$ iff $xzv \in \mathrm{L}(M_L)$. Notice that $|xzv| < n$. By the hypothesis, $xyzv \in \mathrm{L}(M_L)$ iff $xzv \in L$. So, we conclude that $w \in L(M_L)$ iff $w \in L$. □

Example 4.2.5. Let $L = \{a^n b^n \mid i \geq 0\}$ and $L_\# = \#^+ L$. We have shown that $L_\#$ satisfies the pumping condition of Lemma 4.2.1. Now we demonstrate that $L_\#$ does not satisfy the pumping condition of Lemma 4.2.3. Assume the contrary. Let $k \geq 1$ be the constant of Lemma 4.2.3 for $L_\#$. Consider the string $w = \#a^k b^k$ and any decomposition $w = xyz$ such that $y \neq \varepsilon$. If y does not contain the symbol $\#$, that is $y \in (a^+ \cup b^+ \cup a^+ b^+)$, then let $v = \varepsilon$ and, clearly, $wv \in L_\#$ but $xy^2 zv \notin L_\#$. If y contains the symbol $\#$, then let $v = a$ and we have $wv = xyzv \notin L_\#$ but $xzv \in L_\#$. So, $L_\#$ does not satisfy the pumping condition of Lemma 4.2.3. □

Notice that Lemma 4.2.3 requires a decomposition $w = xyz$ that works for all wv, $v \in \Sigma^*$. Another necessary and sufficient pumping lemma for regularity, which does require this type of global condition, was given by Ehrenfeucht, Parikh, and Rozenberg [15]. The latter is called the block pumping lemma and is very similar to Lemma 4.2.2 except that the decomposition of w into xyz has to be along the given division of w into substrings (blocks) $w_1, \ldots, w_k$, so each x, y, and z has to be a catenation of those substrings.

Lemma 4.2.4. *(Block Pumping) $L \subseteq \Sigma^*$ is regular iff there is a constant $k \geq 1$ such that for all $u, v, w \in \Sigma^*$, if $w = w_1 \cdots w_k$, for some $w_1, \ldots, w_k \in \Sigma^*$, then there exist $1 \leq m < n \leq k$ such that $w = xyz$ with $y = w_{m+1} \cdots w_n$, $x, z \in \Sigma^*$, and for all $i \geq 0$, $uwv \in L$ iff $uxy^i zv \in L$.* □

Example 4.2.6. Let $L = \{a^n b^n \mid i \geq 0\}$ and let $L_\$$ be defined as in Example 4.2.4. We have shown in Example 4.2.4 that $L_\$$ satisfies the pumping property of Lemma 4.2.2. Here we show that $L_\$$ does not satisfy the pumping property of Lemma 4.2.4. Assume the contrary. Let k be the constant of the lemma and choose $u = \varepsilon$, $w_1 = \$a$, $w_2 = \$a$, $\dots$, $w_k = \$a$, $v = (\$b)^k$, and $w = w_1 \cdots w_k$. Then, $uwv \in L_\$$. However, clearly, there do not exist m, n, $1 \leq m < n \leq k$, such that $y = w_{m+1} \cdots w_n$, $w = xyz$, and

$$uxzv = uw_1 \cdots w_m w_{n+1} \cdots w_k v = (\$a)^{k-n+m}(\$b^k)\$ \in L_\$,$$

which is a contradiction. □

In Lemma 4.2.4, the pumping condition is sufficient for the regularity of L even if we change the statement „for all $i \geq 0$“ to „for $i = 0$“. Then, the pumping property becomes a cancellation property. It has been shown that the pumping and cancellation properties are equivalent (see [15]). A similar result can be obtained for Lemma 4.2.3.

How to Disprove Context-Freeness

When examining complicated formal languages, we often need to demonstrate that they are non-context-free and, therefore, beyond the power of context-free grammars. The present section based on [38] explains how to make a demonstration like this. We again introduce pumping properties which every context-free language satisfies. Then, we can show that some languages are not context-free by proving that for their sentences these properties do not apply. Unfortunately, these pumping conditions are necessary, however, not sufficient for language to be context-free, so we cannot obtain a positive statement about context-freeness by application of the presented lemma.

The pumping lemma established in this section is frequently used to disprove that a language K is context-free. The lemma says that for every $L \in \mathbf{CF}$, there is a constant $k \geq 1$ such that every $z \in L$ with $|z| \geq k$ can be expressed as $z = uvwxy$ with $vx \neq \varepsilon$ so that L also contains $uv^m wx^m y$, for every $m \geq 0$. Consequently, to demonstrate the non-context-freeness of a language, K, by contradiction, assume that $K \in \mathbf{CF}$ and k is its pumping-lemma constant. Select a string $z \in K$ with $|z| \geq k$, consider all possible decompositions of z into $uvwxy$, and for each of these decompositions, prove that $uv^m wx^m y$ is out of K, for some $m \geq 0$, which contradicts the pumping lemma. Thus, $K \notin \mathbf{CF}$. Without any loss of generality, we prove the pumping lemma based on CFGs satisfying Chomsky normal form (see Definition 2.2.16).

Before we can establish pumping lemma, we need to prove the following theorem concerning a depth of a derivation tree of any sentence of context-free grammar in Chomsky normal form.

Theorem 4.2.1. *Let $G = (V, T, P, S)$ be a CFG in Chomsky normal form. For every derivation $A \Rightarrow^* x$ in G, where $A \in N$ and $x \in T^*$, its corresponding derivation tree $\Delta(A \Rightarrow^* x)$ satisfies $|x| \leq 2^{\mathrm{depth}(\Delta(A \Rightarrow^* x))-1}$.*

Proof. (by induction on $\mathrm{depth}(\Delta(A \Rightarrow^* x)) \geq 1$).

Basis. Let $\mathrm{depth}(\Delta(A \Rightarrow^* x)) = 1$, where $A \in N$ and $x \in T^*$. Because G is in Chomsky normal form, $A \Rightarrow^* x\ [A \to x]$ in G, where $x \in T$, so $|x| = 1$. For $\mathrm{depth}(\Delta(A \Rightarrow^* x)) = 1$, $2^{\mathrm{depth}(\Delta(A \Rightarrow^* x))-1} = 2^0$. As $2^0 = 1$, $|x| \leq 2^{\mathrm{depth}(\Delta(A \Rightarrow^* x))-1}$ in this case, so the basis holds true.

Induction Hypothesis. Suppose that this lemma holds for all derivation trees of depth n or less, for some $n \geq 0$.

Induction Step. Let $A \Rightarrow^* x$ in G with $\text{depth}(\Delta(A \Rightarrow^* x)) = n+1$, where $A \in N$ and $x \in T^*$. Let $A \Rightarrow^* x\ [r\varrho]$ in G, where $r \in P$ and $\varrho \in P^*$. As G is in Chomsky normal form, $r : A \to BC \in P$, where $B, C \in N$. Let $B \Rightarrow^* u\ [\varphi]$, $C \Rightarrow^* v\ [\theta]$, φ, $\theta \in P^*$, $x = uv$, $\varrho = \varphi\theta$ so that $A \Rightarrow^* x$ can be expressed in greater detail as $A \Rightarrow BC\ [r] \Rightarrow^* uC\ [\varphi] \Rightarrow^* uv\ [\theta]$. Observe that $\text{depth}(\Delta(B \Rightarrow^* u\ [\varphi])) \leq \text{depth}(\Delta(A \Rightarrow^* x)) - 1 = n$, so $|u| \leq 2^{\text{depth}(\Delta(B \Rightarrow^* u))-1}$ by the induction hypothesis. Analogously, as $\text{depth}(\Delta(C \Rightarrow^* v\ [\theta])) \leq \text{depth}(\Delta(A \Rightarrow^* x)) - 1 = n$, $|v| \leq 2^{\text{depth}(\Delta(C \Rightarrow^* v))-1}$. Thus, $|x| = |u| + |v| \leq 2^{\text{depth}(\Delta(B \Rightarrow^* u))-1} + 2^{\text{depth}(\Delta(C^* v))-1} \leq 2^{n-1} + 2^{n-1} = 2^n = 2^{\text{depth}(\Delta(A \Rightarrow^* x))-1}$. ☐

Corollary 4.2.1. *Let $G = (V, T, P, S)$ be a CFG in Chomsky normal form. For every derivation $A \Rightarrow^* x$ in G, where $A \in N$ and $x \in T^*$ with $|x| \geq 2^m$ for some $m \geq 0$, its corresponding derivation tree $\Delta(A \Rightarrow^* x)$ satisfies* $\text{depth}(\Delta(A \Rightarrow^* x)) \geq m + 1$.

Proof. This corollary follows from Lemma 4.2.1 and the contrapositive law. ☐

Lemma 4.2.5 (Pumping Lemma for Context-Free Languages). *Let L be an infinite context-free language. Then, there exists $k \geq 1$ such that every string $z \in L$ satisfying $|z| \geq k$ can be expressed as $z = uvwxy$, where $0 < |vx| < |vwx| \leq k$, and $uv^m wx^m y \in L$, for all $m \geq 0$.*

Proof. Let $L \in \mathbf{CF}$, and $L = L(G)$, where $G = (V, T, P, S)$ is a CFG in Chomsky normal form. Let G have n nonterminals, for $n \geq 1$; in symbols, $\text{card}(N) = n$. Set $k = 2^n$. Let $z \in L(G)$ satisfying $|z| \geq k$. As $z \in L(G)$, $S \Rightarrow^* z$, and by Corollary 4.2.1, $\text{depth}(\Delta(S \Rightarrow^* z)) \geq card(N) + 1$, so $\Delta(S \Rightarrow^* z)$ contains some subtrees in which there is a path with two or more nodes labelled by the same nonterminal. Express $S \Rightarrow^* z$ as $S \Rightarrow^* uAy \Rightarrow^+ uvAxy \Rightarrow^+ uvwxy$ with $uvwxy = z$ so that the derivation tree corresponding to $A \Rightarrow^+ vAx \Rightarrow^+ vwx$ contains no proper subtree with a path containing two or more different nodes labelled with the same nonterminal. To prove that $0 < |vx| < |vwx| \leq k$, recall that every rule in P has on its right-hand side either a terminal or two nonterminals because G is in Chomsky normal form. Thus, $A \Rightarrow^+ vAx$ implies $0 < |vx|$, and $vAx \Rightarrow^+ vwx$ implies $|vx| < |vwx|$. As the derivation tree corresponding to $A \Rightarrow^+ vAx \Rightarrow^+ vwx$ contains no subtree with a path containing two different nodes labelled with the same nonterminal, $\text{depth}(\Delta(A \Rightarrow^* vwx)) \leq card(N) + 1$, so by Lemma 4.2.1, $|vx| < |vwx| \leq 2^n = k$. Finally, we demonstrate that for all $m \geq 0$, $uv^m wx^m y \in L$. As $S \Rightarrow^* uAy \Rightarrow^+ uvAxy \Rightarrow^+ uvwxy$, $S \Rightarrow^* uAy \Rightarrow^+ uwy$, so $uv^0 wx^0 y = uwy \in L$. Similarly, since $S \Rightarrow^* uAy \Rightarrow^+ uvAxy \Rightarrow^+ uvwxy$, $S \Rightarrow^* uAy \Rightarrow^+ uvAxy \Rightarrow^+ uvvAxxy \Rightarrow^+ \cdots \Rightarrow^+ uv^m Ax^m y \Rightarrow^+ uv^m wx^m y$, so $uv^m wx^m y \in L$, for all $m \geq 1$. Thus, Lemma 4.2.5 holds true. ☐

We usually use the pumping lemma in a proof by contradiction to demonstrate that a given language L is not context-free. Typically, we make a proof of this kind in the following way.

(1) Assume that L is context-free.

(2) Select a string $z \in L$ whose length depends on the pumping-lemma constant k so that $|z| \geq k$ is necessarily true.

(3) For all possible decompositions of z into $uvwxy$ satisfying the pumping-lemma conditions, find $m \geq 0$ such that $uv^m wx^m y \notin L$, which contradicts Lemma 4.2.5.

(4) The contradiction obtained in (3) means that the assumption in (1) is incorrect; therefore, L is not context-free.

Example 4.2.7. Consider $L = \{a^n b^n c^n \mid n \geq 1\}$. Next, under the guidance of the recommended proof structure preceding this example, we demonstrate that $L \notin \mathbf{CF}$.

(1) Assume that $L \in \mathbf{CF}$.

(2) In L, select $z = a^k b^k c^k$ with $|z| = 3k \geq k$, where k is the pumping-lemma constant.

(3) By Lemma 4.2.5, z can be written as $z = uvwxy$ so that this decomposition satisfies the pumping-lemma conditions. As $0 < |vx| < |vwx| \leq k$, either $vwx \in \{a\}^*\{b\}^*$ or $vwx \in \{b\}^*\{c\}^*$. If $vwx \in \{a\}^*\{b\}^*$, uv^0wx^0y has k cs but fewer than k as or bs, so $uv^0wx^0y \notin L$, but by the pumping-lemma, $uv^0wx^0y \in L$. If $vwx \in \{b\}^*\{c\}^*$, uv^0wx^0y has k as but fewer than k bs or cs, so $uv^0wx^0y \notin L$, but by the pumping lemma, $uv^0wx^0y \in L$. In either case, we obtain the contradiction that $uv^0wx^0y \notin L$ and, simultaneously, $uv^0wx^0y \in L$.

(4) By the contradiction obtained in (3), $L \notin \mathbf{CF}$. □

Omitting some obvious details, we usually proceed in a briefer way than above when proving the non-context-freeness of a language by using Lemma 4.2.5.

Example 4.2.8. Let $L = \{a^n b^m a^n b^m \mid n, m \geq 1\}$. Assume that L is context-free. Set $z = a^k b^k a^k b^k$ with $|a^k b^k a^k b^k| = 4k \geq k$. By Lemma 4.2.5, express $z = uvwxy$. Observe that $0 < |vx| < |vwx| \leq k$ implies $uwy \notin L$ in all possible occurrences of vwx in $a^k b^k a^k b^k$; however, by Lemma 4.2.5, $uwy \in L$—a contradiction. Thus, $L \notin \mathbf{CF}$. □

Even some seemingly trivial unary languages are not context-free as shown next.

Example 4.2.9. Consider $L = \{a^{n^2} \mid \text{for some } n \geq 0\}$. To demonstrate $L \notin \mathbf{CF}$, assume that $L \in \mathbf{CF}$ and select $z = a^{k^2} \in L$ where k is the pumping-lemma constant. As a result, $|z| = k^2 \geq k$, so $z = uvwxy$, which satisfies the pumping-lemma conditions. As $k^2 < |uv^2wx^2y| \leq k^2 + k < k^2 + 2k + 1 = (k+1)^2$, we have $uv^2wx^2y \notin L$, but by Lemma 4.2.5, $uv^2wx^2y \in L$—a contradiction. Thus, $L \notin \mathbf{CF}$. □

How to Disprove Linearity

For general grammars with rules restricted to the linear form (see Definition 2.2.9), which is that each of them contains at most a single nonterminal on the right hand side, we introduce a slightly modified pumping lemma based on the previous pumping lemma for context-free languages; indeed, the languages of these linear grammars represent obviously a bit more restricted language family. The pumping properties shown in the following pumping lemma are necessary but again not sufficient, as demonstrated later, for a language to be linear. Therefore, it can be used only to prove that a certain language is not linear.

Lemma 4.2.6. *Let L be an infinite linear language. Then, there exists $k \geq 1$ such that every string $z \in L$ satisfying $|z| \geq k$ can be expressed as $z = uvwxy$, where*

1. $|vx| > 0$,

2. $|uvxy| \leq k$, and

3. $uv^iwx^iy \in L$, *for all* $i \geq 0$.

Proof. Assume that L is an infinite linear language. Then, there exists a linear grammar $G = (V, T, P, S)$ with $\mathrm{L}(G) = L$, where $|N| = k$, for some $k \geq 1$. Since L is infinite, there exists a string $z \in L$ and a derivation

$$A_0 \Rightarrow_G u_1A_1y_1 \Rightarrow_G u_1u_2A_2y_2y_1 \Rightarrow_G \cdots \Rightarrow_G u_1u_2 \ldots u_ny_n \ldots y_2y_1 = z,$$

where $A_0 = S$, $n \geq k$, $A_l \rightarrow u_{l+1}A_{l+1}y_{l+1} \in P$, for $0 \leq l \leq n-2$, and $A_n \rightarrow u_ny_n \in P$. Then, however, there exist A_i, A_j, where $A_i = A_j$, and $0 \leq i < j \leq k$; informally speaking, since G has a finite number of nonterminals, some of them must necessarily occur repeatedly in a sufficiently long derivation. Let $u = u_1u_2 \cdots u_{i-1}$, $v = u_iu_{i+1} \cdots u_{j-1}$, $w = u_j \cdots y_j$, $x = y_{j-1} \cdots y_{i+1}y_i$, and $y = y_{i-1} \cdots y_2y_1$, so we can express z as $z = uvwxy$. Since A_i and A_j are the same nonterminal

$$\begin{array}{rcll} A_j \Rightarrow_G^* w & \text{implies} & A_i \Rightarrow_G^* w & \text{and} \\ A_i \Rightarrow_G^* vA_jx & \text{implies} & A_j \Rightarrow_G^* vA_jx. & \end{array}$$

Consequently, from $uvwxy \in L$ it follows that $uv^lwx^ly \in L$, for any $l \geq 0$, which completes the proof of Lemma 4.2.6. □

We demonstrate how to disprove linearity by the next example.

Example 4.2.10. Consider the language $L = \{a^mb^mc^nd^n \mid m, n \geq 0\}$. Assume that L is linear. Let $z = a^kb^kc^kd^k$, where $k \geq 1$ is the pumping lemma constant for L. Obviously, $z \in L$ and $|z| \geq k$. Then, $z = uvwxy$, where $|vx| > 0$ and $|uvxy| \leq k$, which implies that $v = a^+$ and/or $x = d^+$. By Lemma 4.2.6, $uv^0wx^0y \in L$, where there are k bs and cs but less than k as and/or ds, so $uv^0wx^0y \notin L$. This is a contradiction and, thus, the assumption does not hold. □

Unfortunately, the presented pumping property of linear languages is not sufficient for a language to be linear as shown in the following example.

Example 4.2.11. Let $L \subseteq \Sigma^*$ be an arbitrary non-linear language and $L_\# = \#^+L\#^+$, where $\# \notin \Sigma$. Then, $L_\#$ satisfies the conditions of Lemma 4.2.6 with the constant k being 2. For any string $w \in \#^+L\#^+$, we can chose $u = y = \varepsilon$ and $v = x = \#$. However, $L_\#$ is not linear, which can be shown as follows. Let h be a morphism defined by $h(a) = a$ for each $a \in \Sigma$ and $h(\#) = \varepsilon$. Then, obviously, $L = h(L_\#)$. Assume that $L_\#$ is linear. Then, L is linear since linear languages are closed under morphism (see [2]). Nevertheless, this contradicts the assumption. Thus, $L_\#$ is not linear. □

How to Disprove Context-Sensitivity

In Section 4.1 we specified workspace space condition that any general grammar must satisfy in order to generate context-sensitive language. We often benefit from them while showing that a certain recursively enumerable language is also context-sensitive. On the other hand, since these conditions are necessary for a language to be context-sensitive, we can also introduce a proof of non-context-sensitivity based on these workspace conditions as stated in Corollary 4.1.1 and demonstrated next.

Example 4.2.12. Consider a language $L = \{w \in \{a\}^* \mid 2^{|w|} + 1 \text{ is prime}\}$. The languages of all strings which length is a prime or a power of 2 are well-known context-sensitive languages (see [14]) which may suggest that also L is a context-sensitive language. However, next we disprove this by showing that no general grammar G with $\mathrm{L}(G) = L$ can satisfy workspace conditions for context-sensitive languages. We present just the idea of the proof since the exhaustive version is beyond the subject of this work.

Assume that there exists a general grammar $G = (V, T, P, S)$ with $\mathrm{L}(G) = L$ satisfying workspace conditions for context-sensitive languages. Then, it generates any $w \in \mathrm{L}(G)$ within $\mathrm{WS}_G(w) \leq k|w|$, for some nonnegative integer k. It is currently not precisely determined what is the time complexity of ideal primality testing, however, it was already shown that it is worst than logarithmic (see [10]). In the sense of general grammars, to test a primality of some $p \geq 1$ grammar G performs a computation through $n = f(p)$ different configurations, where f is a function which grows faster then $\log_2$. Observe that the number of all possible configurations of G is $k|w| \cdot |V|$. Then,

$$k|w| \cdot |V| \geq f(2^{|w|} + 1) > \log_2(2^{|w|} + 1) > \log_2(2^{|w|}) = |w|$$

where k and $|V|$ are constants, so, the expression $k|w| \cdot |V|$ grows linearly with $|w|$. However, the expression $f(2^{|w|} + 1)$ grows faster than linearly with $|w|$ and, therefore, there is some integer $m \geq 1$, where for every $|w| \geq m$

$$k|w| \cdot |V| < f(2^{|w|} + 1)$$

which is a contradiction. As a result, the assumption that there exists a general grammar generating L under the workspace conditions for context sensitive languages is incorrect and the language L is, thus, non-context-sensitive recursively enumerable language. $\square$

Let us only point out that non-context-sensitive recursively enumerable languages are often extremely complicated and these kinds of proofs, thus, very exhaustive and sometimes even a matter of decidability.

Chapter 5

Hierarchy of Language Families

Languages from very simple finite ones through regular, context-free, context-sensitive up to recursively enumerable languages differ enormously in the sense of their structure, decision properties and, of course, the complexity of their description. Some of them can be described in a very simple way—by length of their sentences, composition of symbols, etc.—while the others require extremely complicated formal models to be precisely grasp or are even beyond the decidability. Theory of formal languages always tended to classify them into various families according to their properties and establish hierarchies to properly express differences in their complexity on the rigorous formal basis. Most importantly, in [5] Noam Chomsky first introduced the well-known Chomsky hierarchy of languages and their respective formal models which still stays as an essential formal language classification. Concerning the families of regular, context-free, context-sensitive, and recursively enumerable languages, the next important theorem was stated.

Theorem 5.0.1 (Chomsky Hierarchy, see [5, 6]).

$$\mathbf{REG} \subset \mathbf{CF} \subset \mathbf{CS} \subset \mathbf{RE}$$

Since this topic is vital for the purpose of this work, let us explain the relations between these well-known language families in a greater detail and also cover other important language families which we need to deal with. Therefore, let us now briefly re-establish the proof of Theorem 5.0.1.

Proof. First of all, observe grammars introduced in Section 2.2.

(1) Every context-sensitive grammar is also a general grammar. It holds by the definition.

(2) We previously stated that for every context-free grammar there is a propagating context-free grammar generating the same language. Moreover, every propagating context-free grammar is also a context-sensitive. Indeed, a context-free rule is in fact a context-sensitive rule with a zero-length context.

(3) Every regular grammar is also a context-free grammar. A regular grammar has every rule of one of the forms
$$A \to aB \text{ or } A \to a$$
and both of them are also context free-rules.

From (1) through (3)

$$\mathbf{REG} \subseteq \mathbf{CF} \subseteq \mathbf{CS} \subseteq \mathbf{RE}$$

and we only need to show that these inclusions are in fact proper.

Consider the well-known context free language

$$L = \{a^n b^n \mid n \geq 0\}.$$

By Example 4.2.1, $L \notin \mathbf{REG}$. Thus, $\mathbf{REG} \neq \mathbf{CF}$. Since $\mathbf{REG} \subseteq \mathbf{CF}$, $\mathbf{REG} \subset \mathbf{CF}$.

Consider the following example.

Example 5.0.1. Let $G = (\{S, A, B, C, \bar{C}, a, b, c\}, \{a, b, c\}, P, S)$ be a general grammar with

$$\begin{array}{llllll} P = \{ & 1: S \to ABC, & 2: AB \to AABB\bar{C} & 3: \bar{C}B \to B\bar{C}, & 4: \bar{C}C \to CC \\ & 5: A \to a, & 6: B \to b & 7: C \to c & \}. \end{array}$$

G is obviously monotone and, thus, $\mathrm{L}(G) \in \mathbf{CS}$. Let us investigate a language $\mathrm{L}(G)$.

By the initial rule 1 the string ABC is obtained. Then, by applications of the rule 2 potentially As, Bs, and $\bar{C}$s are added, one of each at the same time. By the rule 3, $\bar{C}$ is moved to the right to be rewritten to C once it occurs by another C. Eventually, all upper-case letters are rewritten to the lower-case ones which completes the derivation. Then,

$$\mathrm{L}(G) = \{a^n b^n c^n \mid n \geq 1\}.$$

A fully rigorous proof is left to the reader. □

Recall Example 4.2.7 which proves that the language $L = \{a^n b^n c^n \mid n \geq 1\}$ is non-context-free. Then, however, $\mathbf{CF} \neq \mathbf{CS}$ and since $\mathbf{CF} \subseteq \mathbf{CS}$, it holds that $\mathbf{CF} \subset \mathbf{CS}$.

In Example 4.2.12 we show that there exist non-context-sensitive recursively enumerable languages. Consequently, $\mathbf{CS} \neq \mathbf{RE}$. Since, $\mathbf{CS} \subseteq \mathbf{RE}$, we obtain $\mathbf{CS} \subset \mathbf{RE}$, and the proof of validity of Chomsky Hierarchy is complete. □

Alongside with the previous well-known language families, we recognize numerous other important families of languages. Some of them also play an important role in the subject of this work and, thus, we investigate their relations to the other language families. First of all, let us analyse the family of finite languages.

Theorem 5.0.2.

$$\mathbf{FIN} \subset \mathbf{REG}$$

Proof. Let $L \subseteq \Sigma^*$ be a finite language. Construct a finite automaton $M = (Q, \Sigma, R, s, F)$ as follows. Let $k = |x|$, where x is the longest string in L. Set

$$Q = \{q_w \mid w \in \Sigma^* \text{ and } |w| \leq k\}$$

and $s = q_\varepsilon$. If $\varepsilon \in L$, put q_ε to F. For every $w \in L$, where $w = a_1 \cdots a_n$, for $a_i \in \Sigma$, $1 \leq i \leq n$, $n \geq 0$, put

1. $q_\varepsilon a_1 \to q_{a_1}$ and
2. $q_{a_1 \cdots a_{j-1}} a_j \to q_{a_1 \cdots a_{j-1} a_j}$ to R, for $2 \leq j \leq n$, and
3. $q_{a_1 \cdots a_n}$ to F.

Then, obviously, for every $w \in L$, where $w = a_1 a_2 \cdots a_n$, for $a_i \in \Sigma$, $0 \leq i \leq n$, $n \geq 0$, we have a computation

$$q_\varepsilon a_1 a_2 \cdots a_n \vdash q_{a_1} a_2 \cdots a_n \vdash \cdots \vdash q_{a_1 a_2 \cdots a_n},$$

where $q_\varepsilon = s$ and $q_{a_1 a_2 \cdots a_n} \in F$. Consequently, $w \in \mathrm{L}(M)$ if $w \in L$. The opposite implication is left to the reader. As a result

$$\mathbf{FIN} \subseteq \mathbf{REG}.$$

To show that the inclusion is in fact proper, consider a general grammar

$$G = (\{A, a\}, \{a\}, \{A \to Aa, A \to a\}, A).$$

By Definition 2.2.10, G is regular and, thus, $\mathrm{L}(G) \in \mathbf{REG}$. Since clearly $\mathrm{L}(G) = a^+$ which is an infinite language, $\mathrm{L}(G) \notin \mathbf{FIN}$. Then, $\mathbf{FIN} \neq \mathbf{REG}$ which completes the proof. □

Next, we show that the family of linear languages is a proper superfamily of the family of regular languages and a proper subfamily of the family of context-free languages.

Theorem 5.0.3.

$$\mathbf{REG} \subset \mathbf{LIN} \subset \mathbf{CF}$$

Proof. By the definitions introduced in Section 2.2, any regular grammar is also a linear grammar and every linear grammar is also a context-free grammar. Therefore,

$$\mathbf{REG} \subseteq \mathbf{LIN} \subseteq \mathbf{CF}$$

and we only need to prove that these inclusions are in fact proper.

Consider the following example.

Example 5.0.2. Let $G = (\{S, a, b\}, \{a, b\}, \{S \to aSb, S \to \varepsilon\}, S)$ be a general grammar. Obviously, G is linear (see Definition 2.2.9) and $\mathrm{L}(G) = \{a^n b^n \mid n \geq 0\}$. □

Let $L = \{a^n b^n \mid n \geq 0\}$. By the previous example, $L \in \mathbf{LIN}$. However, by Example 4.2.1, $L \notin \mathbf{REG}$ and, thus $\mathbf{REG} \neq \mathbf{LIN}$. Consequently,

$$\mathbf{REG} \subset \mathbf{LIN}.$$

Example 5.0.3. Let $G = (\{S, A, B, a, b, c, d\}, \{a, b, c, d\}, P, S)$ be a general grammar, where

$$P = \{S \to AB, A \to aAb, A \to \varepsilon, B \to cBd, B \to \varepsilon\}.$$

Obviously, G is a context-free grammar (see Definition 2.2.9). Let us investigate the language of G. By the initial rule it produces a string AB, from which equally long sequences of as and bs are produced by rewriting A to aAb and equally long sequences of cs and ds are produced by rewriting B to cBd. The derivation once finishes by erasing both nonterminals. Therefore, $\mathrm{L}(G) = \{a^m b^m c^n d^n \mid m, n \geq 0\}$. □

Let $L = \{a^m b^m c^n d^n \mid m, n \geq 0\}$. By the previous example, $L \in \mathbf{CF}$. However, by Example 4.2.10, $L \notin \mathbf{LIN}$ and, thus $\mathbf{LIN} \neq \mathbf{CF}$. Consequently,

$$\mathbf{LIN} \subset \mathbf{CF}$$

which completes the proof of Theorem 5.0.3. □

The last introduced language families plays a rather marginal role in the subject of this work and, so, we do not provide proves of their superiority or inclusion to the other presented language families which the reader may find in the referenced literature.

It was proved that families of context-free languages of finite index form an infinite hierarchy of language families above regular languages and there are also context-free languages of an infinite index—indeed, by observing the definition we can see that ${}_1\mathbf{CF}$ denotes exactly the family of linear languages, while their unlimited versions coincide with the definition of the context-free grammar (for details see [26, 47, 49]).

Theorem 5.0.4.

$$\mathbf{LIN} = {}_1\mathbf{CF} \subset {}_2\mathbf{CF} \subset {}_3\mathbf{CF} \subset \cdots \subset {}_\infty\mathbf{CF} = \mathbf{CF}$$

Matrix grammars first defined and studied in [1] as the very basic concept of controlled rewriting plays an indisputably significant role in the theory of formal languages. Most importantly, it was proved that this abstract family of languages lays in between of the family of context-free and the family of context-sensitive languages. In this study they serve rather as a reference family of languages since it was also proved that they coincide with the family of regular-controlled languages.

Theorem 5.0.5 (see [42]).

$$\mathbf{CF} \subset \mathbf{RC} = \mathbf{MT} \subset \mathbf{CS}$$

Putting together Theorem 5.0.1, 5.0.2, 5.0.3, 5.0.4, and 5.0.5, we obtain the following corollary summarizing the hierarchical relations of all the considered language families.

Corollary 5.0.1.

$$\mathbf{FIN} \subset \mathbf{REG} \subset \mathbf{LIN} = {}_1\mathbf{CF} \subset {}_2\mathbf{CF} \subset \cdots \subset {}_\infty\mathbf{CF} = \mathbf{CF} \subset \mathbf{RC} = \mathbf{MT} \subset \mathbf{CS} \subset \mathbf{RE}$$

Beyond this crucial hierarchical classification of language families there are also languages which cannot be generated by any grammar; as shown next, they in fact represent the vast majority of languages which, unfortunately, corresponds to algorithmically unsolvable problems.

Theorem 5.0.6. *For any alphabet Σ there exists a language $L \subseteq \Sigma^*$, where $L \notin \mathbf{RE}$.*

Proof. For any language $L \in \mathbf{RE}$, $L \subseteq \Sigma^*$, there is a general grammar in Kuroda normal from $G = (V, \Sigma, P, S)$ with $\mathrm{L}(G) = L$ (see Theorem 2.2.3). Since the normal form limits the number of possible rules of P depending on the number of symbols in V, for a certain V we may systematically generate all possible general grammars in Kuroda normal form. Then, we can algorithmically generate all possible general grammars in Kuroda normal form above the given alphabet beginning with those with just one nonterminal symbol and continuously extending the nonterminal alphabet. Consequently, the set of all possible general grammars in Kuroda normal form above the given alphabet is countable. However, since the set of all strings Σ^* above the given alphabet Σ is infinite, the set of all possible languages 2^{Σ^*} is uncountable (follows from Cantor's diagonal argument introduced in [4]). The proof of the theorem, then, follows from the difference in the cardinality of the set of all grammars and the set of all languages above the given alphabet. □

Chapter 6

Tree-Restricted Grammars

In this chapter, we introduce the main results of this thesis. We put simple tree-based conditions for grammars in question and demonstrate that if a grammar satisfy these conditions it in fact generates a context-free language.

6.1 General Grammars

First of all, we focus on the general grammars. As already mentioned (see Section 2.2), they characterize the family of recursively enumerable languages and are, thus, computationally complete. For simplicity, let us consider only general grammars in the binary form (see Definition 2.2.15), since they are equally powerful. In contrast to context-free grammars, general grammars in the binary form can rewrite two neighbouring nonterminals at ones and, thus, perform rewriting of a symbol in context of neighbouring symbols. In the sense of the derivation tree, this introduces two context-dependent nodes occurring in two neighbouring paths; for brevity, we now omit erasing rules. Next, we show that limiting the number of context dependencies between two neighbouring paths decrease the generative power significantly; in fact, precisely to the power of context-free grammars. However, notice that we do not limit the total number of context dependencies at all.

Theorem 6.1.1. *A language L is context-free iff there is a constant $k \geq 0$ and a general grammar G such that $L = \mathrm{L}(G)$ and for every $x \in \mathrm{L}(G)$, there is a tree ${}_G\triangle_x \in {}_G\blacktriangle$ that satisfies:*

1. *any two neighbouring paths contain no more than k pairs of context-dependent nodes;*
2. *out of neighbouring paths, every pair of nodes is context-independent.*

Proof. Construction. Consider any $k \geq 0$. Let $G = (V, T, P, S)$ be a GG such that $\mathrm{L}(G) = L$. Recall, $N = V - T$. Let $P_{cs} \subseteq P$ denote the set of all non-context-free rules of G. Set

$$N' = \{A_{l|r} \mid A \in N,\ l, r \in (P_{cs} \cup \{\varepsilon\})^k\}.$$

Construct a grammar $G' = (V', T, P', S_{\varepsilon|\varepsilon})$, where $V' = N' \cup T$. Set $P' = \emptyset$. Construct P' by performing (I) through (IV) given next.

(I) For all $A \to B \in P$, $A, B \in N$, and $l, r \in (P_{cs} \cup \{\varepsilon\})^k$, add $A_{l|r} \to B_{l|r}$ to P';

(II) for all $A \to a \in P$, $A \in N$, $a \in (T \cup \{\varepsilon\})$, add $A_{\varepsilon|\varepsilon} \to a$ to P';

(III) for all $A \rightarrow BC \in P$, where $A, B, C \in N$, and $r, l, x \in (P_{cs} \cup \{\varepsilon\})^k$, add $A_{l|r} \rightarrow B_{l|x}C_{x|r}$ to P';

(IV) for all $p\colon AB \rightarrow CD \in P$, $A, B, C, D \in N$, $x, z \in (P_{cs} \cup \{\varepsilon\})^k$, and $y \in (P_{cs} \cup \{\varepsilon\})^{k-1}$, add $A_{x|py} \rightarrow C_{x|y}$ and $B_{py|z} \rightarrow D_{y|z}$ to P'.

Basic idea. Notice nonterminal symbols. Since every pair of neighbouring paths of G contains a limited number of context-dependent nodes, all of its context-dependencies are encoded in nonterminals. G' nondeterministically decides about all context-dependencies while introducing a new pair of neighbouring paths by rules (III). A new pair of neighbouring paths is introduced with every application of

$$A_{l|r} \rightarrow B_{l|x}C_{x|r},$$

where x encodes a new descendant context. Context dependencies are realized later by context-free rules (IV).

Since P' contains no non-context-free rule, G' is context-free. Next, we proof $\mathrm{L}(G) = \mathrm{L}(G')$ by establishing Claims 2 through 4. Define the new homomorphism $\gamma : V' \rightarrow V$, $\gamma(A_{l|r}) = A$, for $A_{l,r} \in N'$, and $\gamma(a) = a$ otherwise.

Claim 2. If $S \Rightarrow^m w$ in G, where $m \geq 0$ and $w \in V^*$, then $S_{\varepsilon|\varepsilon} \Rightarrow^* w'$ in G', where $w' \in V'^*$ and $\gamma(w') = w$.

In what follows, for brevity, we sometimes denote a node of derivation tree by the symbol by which it is labelled if there is no risk of confusion.

Proof. We prove this by induction on $m \geq 0$.

Basis. Let $m = 0$. That is $S \Rightarrow^0 S$ in G. Clearly, $S_{\varepsilon|\varepsilon} \Rightarrow^0 S_{\varepsilon|\varepsilon}$ in G', where $\gamma(S_{\varepsilon|\varepsilon}) = S$, so the basis holds.

Induction Hypothesis. Suppose that there exists $n \geq 0$ such that Claim 2 holds for all m with $0 \leq m \leq n$.

Induction Step. Let $S \Rightarrow^{n+1} w$ in G. Then, $S \Rightarrow^n v \Rightarrow w$, where $v \in V^*$, and there exists $p \in P$ such that $v \Rightarrow w\,[p]$. By the induction hypothesis, $S_{\varepsilon|\varepsilon} \Rightarrow^* v'$, where $\gamma(v') = v$, in G'. Next, we consider the following four forms of p.

(I) Let $p\colon A \rightarrow B \in P$, for some $A, B \in N$. Without any loss of generality, suppose l and r are a left descendant context and a right descendant context of A. By the construction of G', there exists a rule $p'\colon A_{l|r} \rightarrow B_{l|r} \in P'$. Then, there exists a derivation $v' \Rightarrow w'\,[p']$ in G', where $\gamma(w') = w$.

(II) Let $p\colon A \rightarrow a \in P$, for some $A \in N$ and $a \in T \cup \{\varepsilon\}$. Since a is a terminal symbol, it corresponds to a node with empty descendant contexts. By the construction of G', there exists a rule $p'\colon A_{\varepsilon|\varepsilon} \rightarrow a \in P'$. Then, there exists a derivation $v' \Rightarrow w'\,[p']$ in G', where $\gamma(w') = w$.

(III) Let $p\colon A \rightarrow BC \in P$, for some $A, B, C \in N$. Without any loss of generality, suppose l and r are a left descendant context and a right descendant context of A, and $x \in (P_{cs} \cup \{\varepsilon\})^k$ is a context of neighbouring paths beginning at this node. By the construction of G', there exists a rule $p'\colon A_{l|r} \rightarrow B_{l|x}C_{x|r} \in P'$. Then, there exists a derivation $v' \Rightarrow w'\,[p']$ in G', where $\gamma(w') = w$.

(IV) Let $p\colon AB \to CD \in P$, for some $A, B, C, D \in N$. By the assumption stated in Theorem 6.1.1, A and B occur in two neighbouring paths denoted by α and β, respectively. Without any loss of generality, suppose that a context of α and β is a string $c \in (P_{cs} \cup \varepsilon)^k$, where $c = pc_f$, and l is a left descendant context, r is a right descendant context of A, B, respectively. By the construction of G', there exist two rules

$$p'_l\colon A_{l|pc_f} \to C_{l|c_f}, \quad p'_r\colon B_{pc_f|r} \to D_{c_f|r} \in P'.$$

Then, there exists a derivation $v' \Rightarrow^2 w'\ [p'_l p'_r]$ in G', where $\gamma(w') = w$.

Notice ((IV)). The preservation of the context is achieved by nonterminal symbols. Since the stored context is reduced symbol by symbol from left to right direction in both α and β, G' simulates the applications of non-context-free rules of G.

We covered all possible forms of p, so the claim holds. □

Claim 3. Every $x \in \mathscr{F}(G')$ can be derived in G' as follows.

$$S_{\varepsilon|\varepsilon} = x_0 \Rightarrow^{d_1} x_1 \Rightarrow^{d_2} x_2 \Rightarrow^{d_3} \cdots \Rightarrow^{d_{h-1}} x_{h-1} \Rightarrow^{d_h} x_h = x,$$

for some $h \geq 0$, where $d_i \in \{1, 2\}$, $1 \leq i \leq h$, so that

1. *if $d_i = 1$, then $x_{i-1} = uA_{l|r}v$, $x_i = uzv$, $x_{i-1} \Rightarrow x_i\ [A_{l|r} \to z]$, where $u, v \in V'^*$, $z \in \{B_{l|r}, C_{l|x}D_{x|r}, a\}$, for some $A_{l|r}, B_{l|r}, C_{l|x}, D_{x|r} \in N'$, $a \in (T \cup \{\varepsilon\})$;*

2. *if $d_i = 2$, then $x_{i-1} = uA_{x|py}B_{py|z}v$, $x_i = uC_{x|y}D_{y|z}v$, and*

$$uA_{x|py}B_{py|z}v \Rightarrow uC_{x|y}B_{py|z}v\ [A_{x|py} \to C_{x|y}] \Rightarrow uC_{x|y}D_{y|z}v\ [B_{py|z} \to D_{y|z}],$$

for some $u, v \in V'^*$ and $A_{x|py}, B_{py|z}, C_{x|y}, D_{y|z} \in N'$.

Proof. Since G' is context-free, without any loss of generality in every derivation of G' we can always reorder applied rules to satisfy Claim 3. □

Claim 4. Let $S_{\varepsilon|\varepsilon} \Rightarrow^{d_1} x_1 \Rightarrow^{d_2} \cdots \Rightarrow^{d_{m-1}} x_{m-1} \Rightarrow^{d_m} x_m$ in G' be a derivation that satisfies Claim 3, for some $m \geq 0$. Then, $S \Rightarrow^* w$ in G, where $\gamma(x_m) = w$.

Proof. We prove this by induction on $m \geq 0$.

Basis. Let $m = 0$. That is $S_{\varepsilon|\varepsilon} \Rightarrow^0 S_{\varepsilon|\varepsilon}$ in G'. Clearly, $S \Rightarrow^0 S$ in G. Since $\gamma(S_{\varepsilon|\varepsilon}) = S$, the basis holds.

Induction Hypothesis. Suppose that there exists $n \geq 0$ such that Claim 4 holds for all m with $0 \leq m \leq n$.

Induction Step. Let $S_{\varepsilon|\varepsilon} \Rightarrow^{d_1} x_1 \Rightarrow^{d_2} \cdots \Rightarrow^{d_{n-1}} x_{n-1} \Rightarrow^{d_n} x_n \Rightarrow^{d_{n+1}} x_{n+1}$ in G' be a derivation that satisfies Claim 3. By the induction hypothesis, $S \Rightarrow^* v$, $v \in V^*$, where $\gamma(x_n) = v$, in G. Divide the proof into two parts according to d_{n+1}.

(A) Let $d_{n+1} = 1$. By the construction of G', there exists a rule $p' \in P'$ such that $x_n \Rightarrow^{d_{n+1}} x_{n+1}\ [p']$. Next, we consider the following three forms of p'.

(I) Let $p'\colon A_{l|r} \to B_{l|r} \in P'$, for some $A, B \in N$ and $l, r \in (P_{cs} \cup \{\varepsilon\})^k$. By the construction of G', rule p' is introduced by some rule $p\colon A \to B \in P$. Then, there exists a derivation $v \Rightarrow w\,[p]$, where $\gamma(x_{n+1}) = w$.

(II) Let $p'\colon A_{\varepsilon|\varepsilon} \to a \in P'$, for some $A \in N$ and $a \in T \cup \{\varepsilon\}$. By the construction of G', rule p' is introduced by some rule $p\colon A \to a \in P$. Then, there exists a derivation $v \Rightarrow w\,[p]$, where $\gamma(x_{n+1}) = w$.

(III) Let $p'\colon A_{l|r} \to B_{l|x}C_{x|r} \in P'$, for some $A, B, C \in N$ and $l, r, x \in (P_{cs} \cup \{\varepsilon\})^k$. By the construction of G', rule p' is introduced by some rule $p\colon A \to BC \in P$. Then, there exists a derivation $v \Rightarrow w\,[p]$, where $\gamma(x_{n+1}) = w$.

(B) Let $d_{n+1} = 2$. Then, $x_n \Rightarrow^{d_{n+1}} x_{n+1}$ is equivalent to

$$u_1A_{x|py}B_{py|z}u_2 \Rightarrow u_1C_{x|y}B_{py|z}u_2\ [p'_1] \Rightarrow u_1C_{x|y}D_{y|z}u_2\ [p'_2],$$

where $x_n = u_1A_{x|py}B_{py|z}u_2$, $x_{n+1} = u_1C_{x|y}D_{y|z}u_2$, and

$$p'_1\colon A_{x|py} \to C_{x|y},\ p'_2\colon B_{py|z} \to D_{y|z} \in P',$$

for some $u_1, u_2 \in V'^*$ and $A_{x|py}$, $B_{py|z}$, $C_{x|y}$, $D_{y|z} \in N'$. By the construction of G', rules p'_1 and p'_2 were introduced by some rule $p\colon AB \to CD \in P$, Then, there exists a derivation $v \Rightarrow w\,[p]$, where $\gamma(x_{n+1}) = w$.

We covered all possibilities, so the claim holds. □

By Claims 2 and 4, $S \Rightarrow^* w$ in G iff $S_{\varepsilon|\varepsilon} \Rightarrow^* w'$ in G', where $\gamma(w') = w$. If $S \Rightarrow^* w$ in G and $w \in T^*$, then $w \in \mathrm{L}(G)$. Since $\gamma(w') = w' = w$, for $w \in T^*$, $w' \in \mathrm{L}(G')$. Therefore, $\mathrm{L}(G) = \mathrm{L}(G')$ and Theorem 6.1.1 holds. □

Consider Theorem 6.1.1. Observe that the second condition is superfluous whenever G is monotone. Since a grammar is in the binary form and no symbol can be erased, all context dependencies are within pairs of neighbouring paths.

Theorem 6.1.2. *A language L is context-free iff there is a constant $k \geq 0$ and a monotone general grammar G such that $L = \mathrm{L}(G)$ and for every $x \in \mathrm{L}(G)$, there is a tree ${}_G\triangle_x \in {}_G\blacktriangle$, where any two neighbouring paths contain no more than k pairs of context-dependent nodes.*

Proof. Prove this by analogy with the proof of Theorem 6.1.1. □

We proved that Theorems 6.1.1 and 6.1.2 introduce necessary but also sufficient conditions for a general grammar to generate context-free language. Later in Chapter 7 we demonstrate how to use this result to obtain a positive proof of context-freeness of a language in practice.

6.2 Regular-Controlled Grammars

Now we turn our attention to regulated grammars; namely, grammars regulated by regular control languages over the set of rules called regular-controlled grammars (see Section 2.2). They were introduced in [19] and are as powerful as matrix grammars, so, they define the family of matrix languages. Every RCG consists of a context-free grammar G and a regular language C above the rules of G and the sentences in C define the only valid derivations of G. Therefore, this control mechanism ensures a certain order of the applied rules. Let us look at it from the derivation-tree point of view. Even an ordinary context-free grammar can introduce rule-application synchronization based on nonterminal symbols within a single branch of the derivation tree. On top of this, RCGs can synchronize the applications of rules also in the case when the path of the derivation changes a branch of the derivation tree. However, what if we limit the number of possible path changes by a constant? The present section demonstrates that the language of an RCG limited in this way is context-free. Moreover, for a limiting constant k, the language is in fact context-free of index $k+1$, where, additionally, this index is only an upper bound. As a result, this restriction of the derivation trees of RCGs is even more restrictive than we originally estimated.

Theorem 6.2.1. *If there is a constant $k \geq 0$ and a regular-controlled grammar H in binary form such that, for every $w \in L(H)$, there exists a derivation of w in H with at most k path-changing derivation steps, then $L(H)$ is a context-free language, and moreover, it is of index $k+1$.*

An RCG satisfying restriction from Theorem 6.2.1 is said to be *k-restricted.*

Proof. Let $\bar{H} = (\bar{G}, \bar{C})$, $\bar{G} = (\bar{V}, T, \bar{P}, S)$, be an RCG in the binary form such that $\mathrm{L}(\bar{H}) = L$ and let $k \geq 0$ be a constant such that for every $x \in \mathrm{L}(\bar{H})$, there exists a derivation $S \Rightarrow^* x$ in $\bar{H}$ with k or fewer path-changing derivation steps.

Preliminary transformation. Construct $H = (G, C)$, $G = (V, T, P, S)$, as follows. Initially, set $C = \emptyset$, $V = \bar{V}$, and $P = \{r \mid r\colon A \to w \in \bar{P}, \#_{\bar{N}}(w) = 1\}$. Define the new homomorphism h over P as $h(x) = x$, for all $x \in P$. For every rule $r\colon A \to BC \in \bar{P}$, where $A, B, C \in \bar{N}$, add new nonterminal $\langle r \rangle$ to N and two new rules

$$r_1\colon A \to \langle r \rangle C, \;\; r_2\colon \langle r \rangle \to B$$

to P and redefine h so that $h(r) = r_1 r_2$. For every rule $r\colon A \to w \in \bar{P}$, where $A \in \bar{N}$ and $w \in T^*$, add new nonterminal $\langle r \rangle$ to N and two new rules

$$r_1\colon A \to \langle r \rangle, \;\; r_2\colon \langle r \rangle \to w$$

to P and redefine h so that $h(r) = r_1 r_2$. Finally set $C = h(\bar{C})$.

Claim 5. $\mathrm{L}(\bar{H}) = \mathrm{L}(H)$.

Proof. Since H is constructed so that every rule of the form $r\colon A \to BC$ or $r\colon A \to w$ is substituted by two always consecutively applied rules $r_1\colon A \to \langle r \rangle C$ and $r_2\colon \langle r \rangle \to B$ or $r_1\colon A \to \langle r \rangle$ and $r_2\colon \langle r \rangle \to w$, respectively, working equally, it is obvious that the claim holds. □

Moreover, the preliminary transformation does not add any new branching and, thus, preserves k as a valid limit of path-changes.

The previous transformation aims to simplify the next construction proof. We avoid path-changes during branching and directly before leafs. Additionally, after every branching the derivation always continues with the left child node.

Construction. Let $M = (Q, P, R, s, F)$ be a finite automaton such that $\mathrm{L}(M) = C$. Set

$$\begin{gathered}\bar{N} = \{\langle \bar{A}|q|r|s|t|f\rangle \mid A \in N; q \in Q; r,s,t \in (Q \cup \{\varepsilon\})^k; f \in F \cup \{\varepsilon\}\},\\ N' = \{\langle A|q|r|s|t|f\rangle \mid A \in N; q \in Q; r,s,t \in (Q \cup \{\varepsilon\})^k,\\ f \in F \cup \{\varepsilon\}\} \cup \{\langle S'|s|\varepsilon|\varepsilon|\varepsilon|\varepsilon\rangle\} \cup \bar{N},\end{gathered}$$

where $S' \notin N$. Set $V' = N' \cup T$. Construct a context-free grammar $G' = (V', T, P', \langle S'|s|\varepsilon|\varepsilon|\varepsilon|\varepsilon\rangle)$. Set $P' = \emptyset$. Construct P' by performing (I) through (VI) given next.

(I) For all $x \in (Q \cup \{\varepsilon\})^k$ and $f \in F$, add $\langle S'|s|\varepsilon|\varepsilon|\varepsilon|\varepsilon\rangle \rightarrow \langle S|s|\varepsilon|\varepsilon|x|f\rangle$ to P';

(II) for all $r\colon A \rightarrow uBv \in P$, $qr \vdash p \in R$, $x, y, z \in (Q \cup \{\varepsilon\})^k$, and $f \in F \cup \{\varepsilon\}$, where $B \in N$, $uv \in T^*$, add

- (i) $\langle A|q|x|y|z|f\rangle \rightarrow u\langle B|p|x|y|z|f\rangle v$,
- (ii) $\langle \bar{A}|q|x|y|z|f\rangle \rightarrow u\langle B|p|x|y|z|f\rangle v$ to P';

(III) for all $r\colon A \rightarrow uBv \in P$, $qr \vdash p \in R$, $g \in Q$, $x, y, z \in (Q \cup \{\varepsilon\})^k$, and $f \in F \cup \{\varepsilon\}$, where $B \in N$, $uv \in T^*$, add $\langle A|g|gx|qy|z|f\rangle \rightarrow u\langle B|p|x|y|z|f\rangle v$ to P';

(IV) for all $r\colon A \rightarrow w \in P$ and $qr \vdash p \in Q$, where $w \in T^*$, add

- (i) $\langle A|q|p|\varepsilon|\varepsilon|\varepsilon\rangle \rightarrow w$,
- (ii) $\langle \bar{A}|q|p|\varepsilon|\varepsilon|\varepsilon\rangle \rightarrow w$ to P';

(V) for all $r\colon A \rightarrow w \in P$ and $qr \vdash f \in Q$, where $w \in T^*$, $f \in F$, add

- (i) $\langle A|q|\varepsilon|\varepsilon|\varepsilon|f\rangle \rightarrow w$,
- (ii) $\langle \bar{A}|q|\varepsilon|\varepsilon|\varepsilon|f\rangle \rightarrow w$ to P';

(VI) for all $r\colon A \rightarrow BC \in P$, $qr \vdash p \in R$, $B, C \in N$, $g \in Q$, $x_1x_2, y_1y_2, z_1z_2z_3z_4 \in (Q \cup \{\varepsilon\})^k$, $f \in F \cup \{\varepsilon\}$, $f_1f_2 = f$, add

- (i) $\langle A|q|\operatorname{shuffle}(x_1,x_2)|\operatorname{shuffle}(y_1,y_2)|gz_1z_2z_3z_4|f\rangle \rightarrow$ $\langle B|p|\operatorname{shuffle}(x_1,gz_1)|\operatorname{shuffle}(y_1,z_2)|z_3|f_1\rangle\langle \bar{C}|g|\operatorname{shuffle}(x_2,z_2)|\operatorname{shuffle}(y_2,z_1)|z_4|f_2\rangle$,
- (ii) $\langle \bar{A}|q|\operatorname{shuffle}(x_1,x_2)|\operatorname{shuffle}(y_1,y_2)|gz_1z_2z_3z_4|f\rangle \rightarrow$ $\langle B|p|\operatorname{shuffle}(x_1,gz_1)|\operatorname{shuffle}(y_1,z_2)|z_3|f_1\rangle\langle \bar{C}|g|\operatorname{shuffle}(x_2,z_2)|\operatorname{shuffle}(y_2,z_1)|z_4|f_2\rangle$,
- (iii) $\langle A|q|\operatorname{shuffle}(x_1,x_2)|g\operatorname{shuffle}(y_1,y_2)|z_1z_2z_3z_4|f\rangle \rightarrow$ $\langle B|p|\operatorname{shuffle}(x_1,z_1)|\operatorname{shuffle}(y_1,z_2)|z_3|f_1\rangle\langle \bar{C}|g|\operatorname{shuffle}(x_2,z_2)|\operatorname{shuffle}(y_2,z_1)|z_4|f_2\rangle$,
- (iv) $\langle \bar{A}|q|\operatorname{shuffle}(x_1,x_2)|g\operatorname{shuffle}(y_1,y_2)|z_1z_2z_3z_4|f\rangle \rightarrow$ $\langle B|p|\operatorname{shuffle}(x_1,z_1)|\operatorname{shuffle}(y_1,z_2)|z_3|f_1\rangle\langle \bar{C}|g|\operatorname{shuffle}(x_2,z_2)|\operatorname{shuffle}(y_2,z_1)|z_4|f_2\rangle$ to P'.

Define the new morphism $\gamma\colon V'^* \rightarrow V^*$ such that for $\langle A|q|x|y|z|f\rangle \in N'$, $\gamma(\langle A|q|x|y|z|f\rangle) = A$, $\gamma(x) = x$ otherwise.

Basic idea. The context-free grammar G' is designed to simulate the derivations of H. Since in any derivation of H there are k or fewer path-changes, G' nondeterministically decides about all the path-changes during the initial derivation step. To satisfy the restrictions given by control language C, the automaton M, $\mathrm{L}(M) = L$, is encoded in the rules of G'. While performing linear derivations, the consecutivity of states is ensured. When a new branching node is introduced, it is nondeterministically decided about path-changes between both subtrees of the derivation tree which are encoded in nonterminals and simulated by context free rules.

Let us describe the composite nonterminal symbols in greater detail. For a symbol

$$\langle A|q|x|y|z|f\rangle$$

composed of symbol A, states q and f, and the stings of zero up to k states x, y, and z, we refer to A, q, x, y, z, and f as the first, second, third, fourth, fifth, and sixth component, respectively. The first component encodes nonterminal symbol itself, while the others encode states of the finite automaton M with $L(M) = C$. The second component encodes the current state of M. The third component holds the string of states from which there is a path-change underneath the current branch of the derivation tree, while the fourth component holds the string of states into which there is a path-change underneath the current branch of the derivation tree. The fifth component represents a string of branching states, which are to be set as the branching ones during the rest of the derivation. Finally, the sixth component encodes the final state of M to be reached.

Let us informally describe six classes of the rules of G':

(I) An initial rule of the form $\langle S'|s|\varepsilon|\varepsilon|\varepsilon|\varepsilon\rangle \rightarrow \langle S|s|\varepsilon|\varepsilon|x|f\rangle$ rewriting the start symbol is applied only once at the beginning of any derivation. It nondeterministically generates x—a string of all states in which there is a path-change—and f—a final state of M to be reached—which are then saved in the fifth and sixth component of a nonterminal, respectively.

(II) The rules of the form $\langle A|q|x|y|z|f\rangle \rightarrow u\langle B|p|x|y|z|f\rangle v$ simulate consecutive path-preserving linear derivations which are designed to follow transitions in M or path-changes into the right child of a new branching node. The first component of a nonterminal represents nonterminal in G, while the second represents a state of M.

(III) The rules of the form $\langle A|g|gx|qy|z|f\rangle \rightarrow u\langle B|p|x|y|z|f\rangle v$ represent path-changes. Since the third component of a nonterminal represents a string of states in which the path-changes out of the subtree of the current node occur, a path-change may be performed only when the first symbol corresponds to the current state of M. The fourth component represent a string of states in which the path-changes into the subtree of the current node occur. Since there is a path-change out and the node is not terminal, if the derivation is successful, there once follows a path-change back simulated by the rule.

Notice that the rules (III) cannot rewrite noterminals from $\bar{N}$ generated by the rules (VI). They simulate path-change out and the following path-change back at once which, however, does not correspond to a path-change into the right subbranch of a new branching node.

(IV) The rules of the form $\langle A|q|p|\varepsilon|\varepsilon|\varepsilon\rangle \rightarrow w$ act slightly similarly to (III), however, a new node is terminal and M does not terminate yet, thus, a path-change out of the current branch must be performed, but there is no path-change back. Additionally, all the previously nondeterministically planed path-changes must be already done—fourth and sixth component of a nonterminal is empty and the third contains precisely one state.

(V) A rule of the form $\langle A|q|\varepsilon|\varepsilon|\varepsilon|f\rangle \rightarrow w$ terminates the current derivation with respect to M, therefore, there follows no path-change. In every successful derivation there is always exactly one such rule applied.

(VI) The last class of the rules represents branching. Since G is in the binary form, every node has at most two children. Moreover, by the preliminary transformation of H it is ensured that the derivation follows by rewriting the left newly introduced nonterminal, thus, we do not consider other cases (e.g. path-changing while branching). To terminate the right branch, there must once occur a path-change into it which is planed while branching. A path-change may lead from the subtree of the left branch—(i)-(ii)—or is already planed—(iii)-(iv).

The third and fourth components of $\langle A|q|\operatorname{shuffle}(x_1, x_2)|\operatorname{shuffle}(y_1, y_2)|z_1z_2z_3z_4|f\rangle$ are nondeterministically divided into newly introduced branches, but the mutual order of the states is preserved, and some path-changes from the fifth component may be nondeterministically planed between both new branches. Finally, if $f \in F$, it is decided into which branch it is put.

We note that there is a lot of rules or nonterminals which possibly do not occur in any successful derivation. Moreover, a nondeterministic generation and distribution of path-changing states may result into blocking of a derivation. As we prove next, this, however, does not change the language of the grammar.

However, before we complete the proof of Theorem 6.2.1, let us clarify the construction part of it by providing the following illustrative example.

Example 6.2.1. Consider RCG $H = (G, C)$ from Example 3.2.5 and let $k = 1$. Recall $G = (\{S, A, B, a, b, c, d, e, f\}, \{a, b, c, d, e, f\}, P, S)$, $C = \{1\}^*\{2\}\{3\}^*\{4\}\{5\}^*\{6\}$, and

$$\begin{aligned} P = \{\ & 1\colon S \to aSb, && 2\colon S \to AB, \\ & 3\colon A \to cAd, && 4\colon A \to \varepsilon, \\ & 5\colon B \to eBf, && 6\colon B \to \varepsilon\ \}. \end{aligned}$$

Construct $H' = (\bar{G}, \bar{C})$ according to the preliminary transformation of the proof of Theorem 6.2.1 with $\bar{G} = (\{S, A, B, \langle 2\rangle, \langle 4\rangle, \langle 6\rangle, a, b, c, d, e, f\}, \{a, b, c, d, e, f\}, \bar{P}, S)$,

$$\begin{aligned} \bar{P} = \{\ & 1\colon S \to aSb, && 2_1\colon S \to \langle 2\rangle B, && 2_2\colon \langle 2\rangle \to A, \\ & 3\colon A \to cAd, && 4_1\colon A \to \langle 4\rangle, && 4_2\colon \langle 4\rangle \to \varepsilon, \\ & 5\colon B \to eBf, && 6_1\colon B \to \langle 6\rangle, && 6_2\colon \langle 6\rangle \to \varepsilon\ \}, \end{aligned}$$

and $\bar{C} = \{1\}^*\{2_1\}\{2_2\}\{3\}^*\{4_1\}\{4_2\}\{5\}^*\{6_1\}\{6_2\}$. Define an FA $M = \{\{s, s_q, q, q_p, p, p_f, f\}, \{1, 2_1, 2_2, 3, 4_1, 4_2, 5, 6_1, 6_2\}, R, s, \{f\}\}$ with

$$\begin{aligned} R = \{\ & s1 \vdash s, && s2_1 \vdash s_q, && s_q2_2 \vdash q, \\ & q3 \vdash q, && q4_1 \vdash q_p, && q_p4_2 \vdash p, \\ & p5 \vdash p, && p6_1 \vdash p_f, && p_f6_2 \vdash f\ \}. \end{aligned}$$

Next, we define a CFG simulating H', however, to make it as readable as possible, we list only essential nonterminals and rules; despite this example is quite simple, the grammar contains thousands of them, but only very few nonterminals are reachable and terminating (see [38]) and very few rules applicable in any derivation. Define

$$G' = (V', \{a, b, c, d, e, f\}, P', \langle S'|s|\varepsilon|\varepsilon|\varepsilon|\varepsilon\rangle)$$

with

$$\begin{aligned} V' = \{ & \langle S'|s|\varepsilon|\varepsilon|\varepsilon|\varepsilon\rangle, \langle S|s|\varepsilon|\varepsilon|p|f\rangle, \langle\langle 2\rangle|s_q|p|\varepsilon|\varepsilon|\varepsilon\rangle, \langle B|p|\varepsilon|\varepsilon|\varepsilon|f\rangle, \\ & \langle A|q|p|\varepsilon|\varepsilon|\varepsilon\rangle, \langle\langle 4\rangle|q_p|p|\varepsilon|\varepsilon|\varepsilon\rangle, \langle\langle 6\rangle|p_f|\varepsilon|\varepsilon|\varepsilon|f\rangle\} \cup \{a, b, c, d, e, f\} \end{aligned}$$

$$
\begin{aligned}
P' = \{\ & \dot{0}\colon \langle S'|s|\varepsilon|\varepsilon|\varepsilon\rangle && \to \langle S|s|\varepsilon|\varepsilon|p|f\rangle,\\
& \dot{1}\colon \langle S|s|\varepsilon|\varepsilon|p|f\rangle && \to a\langle S|s|\varepsilon|\varepsilon|p|f\rangle b,\\
& \dot{2}\colon \langle S|s|\varepsilon|\varepsilon|p|f\rangle && \to \langle\langle 2\rangle|s_q|p|\varepsilon|\varepsilon|\varepsilon\rangle\langle B|p|\varepsilon|\varepsilon|\varepsilon|f\rangle,\\
& \dot{3}\colon \langle\langle 2\rangle|s_q|p|\varepsilon|\varepsilon|\varepsilon\rangle && \to \langle A|q|p|\varepsilon|\varepsilon|\varepsilon\rangle,\\
& \dot{4}\colon \langle A|q|p|\varepsilon|\varepsilon|\varepsilon\rangle && \to c\langle A|q|p|\varepsilon|\varepsilon|\varepsilon\rangle d,\\
& \dot{5}\colon \langle A|q|p|\varepsilon|\varepsilon|\varepsilon\rangle && \to \langle\langle 4\rangle|q_p|p|\varepsilon|\varepsilon|\varepsilon\rangle,\\
& \dot{6}\colon \langle\langle 4\rangle|q_p|p|\varepsilon|\varepsilon|\varepsilon\rangle && \to \varepsilon,\\
& \dot{7}\colon \langle B|p|\varepsilon|\varepsilon|\varepsilon|f\rangle && \to e\langle B|p|\varepsilon|\varepsilon|\varepsilon|f\rangle f,\\
& \dot{8}\colon \langle B|p|\varepsilon|\varepsilon|\varepsilon|f\rangle && \to \langle\langle 6\rangle|p_f|\varepsilon|\varepsilon|\varepsilon|f\rangle,\\
& \dot{9}\colon \langle\langle 6\rangle|p_f|\varepsilon|\varepsilon|\varepsilon|f\rangle && \to \varepsilon,\\
& \qquad\qquad \dots\\
\}&.
\end{aligned}
$$

For easier referencing, we add a unique label to each rule. Consider the derivation 123456 in G from Example 3.2.5. The corresponding derivation in G' is as follows.

$$
\begin{aligned}
\langle S'|s|\varepsilon|\varepsilon|\varepsilon\rangle &\Rightarrow \langle S|s|\varepsilon|\varepsilon|p|f\rangle && [\dot{0}]\\
&\Rightarrow a\langle S|s|\varepsilon|\varepsilon|p|f\rangle b && [\dot{1}]\\
&\Rightarrow a\langle\langle 2\rangle|s_q|p|\varepsilon|\varepsilon|\varepsilon\rangle\langle B|p|\varepsilon|\varepsilon|\varepsilon|f\rangle b && [\dot{2}]\\
&\Rightarrow a\langle A|q|p|\varepsilon|\varepsilon|\varepsilon\rangle\langle B|p|\varepsilon|\varepsilon|\varepsilon|f\rangle b && [\dot{3}]\\
&\Rightarrow ac\langle A|q|p|\varepsilon|\varepsilon|\varepsilon\rangle d\langle B|p|\varepsilon|\varepsilon|\varepsilon|f\rangle b && [\dot{4}]\\
&\Rightarrow ac\langle\langle 4\rangle|q_p|p|\varepsilon|\varepsilon|\varepsilon\rangle d\langle B|p|\varepsilon|\varepsilon|\varepsilon|f\rangle b && [\dot{5}]\\
&\Rightarrow acd\langle B|p|\varepsilon|\varepsilon|\varepsilon|f\rangle b && [\dot{6}]\\
&\Rightarrow acde\langle B|p|\varepsilon|\varepsilon|\varepsilon|f\rangle fb && [\dot{7}]\\
&\Rightarrow acde\langle\langle 6\rangle|p_f|\varepsilon|\varepsilon|\varepsilon|f\rangle fb && [\dot{8}]\\
&\Rightarrow acdefb && [\dot{9}]
\end{aligned}
$$

However, it also corresponds to $s12_12_234_14_256_16_2 \vdash^* f$ in M and, thus, to H'. Notice step $\dot{0}$, where $\langle S|s|\varepsilon|\varepsilon|p|f\rangle$ is generated. It encodes that the grammar must once simulate a path-change in state p and apply terminating rule entering final state f—which is step $\dot{9}$—with respect to M. In branching step $\dot{2}$, state p is put to the third component of $\langle\langle 2\rangle|s_q|p|\varepsilon|\varepsilon|\varepsilon\rangle$ which encodes that it once must be reached in the left branch—this is done in step $\dot{6}$—and to the second component of $\langle B|p|\varepsilon|\varepsilon|\varepsilon|f\rangle$ simulating that the derivation continues from the same state with respect to M. Fig. 6.2.1 demonstrates how G' follows M.

□

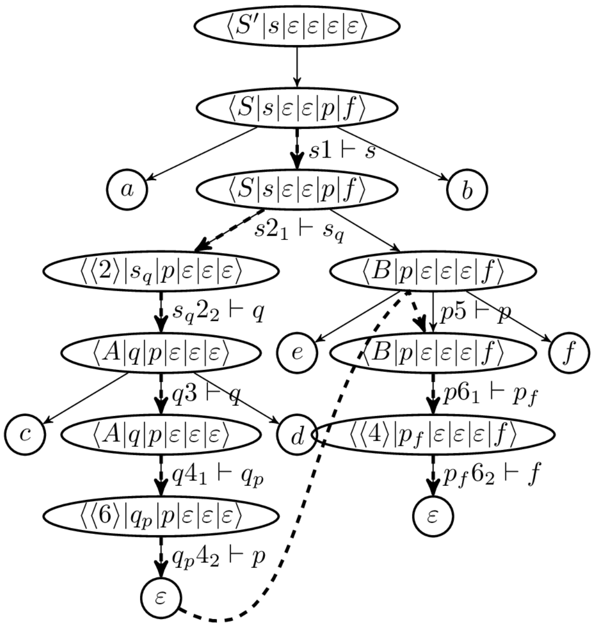
Claim 6. If $S \Rightarrow^m w$ in H, where $m \geq 0$ and $w \in V^*$, then $\langle S'|s|\varepsilon|\varepsilon|\varepsilon\rangle \Rightarrow^* w'$ in G', where $w' \in V'^*$ and $\gamma(w') = w$.

Proof. We prove the statement by induction on $m \geq 0$.

Basis. Let $m = 0$. That is, $S \Rightarrow^0 S$ in H. Clearly, $\langle S'|s|\varepsilon|\varepsilon|\varepsilon\rangle \Rightarrow \langle S|s|\varepsilon|\varepsilon|x|f\rangle$ in G', where $\gamma(\langle S|s|\varepsilon|\varepsilon|x|f\rangle) = S$, for some $x \in (Q \cup \{\varepsilon\})^k$ and $f \in F$, so the basis holds.

Induction Hypothesis. Suppose that there exists $n \geq 0$ such that Claim 6 holds for all m with $0 \leq m \leq n$.

Induction Step. Let $S \Rightarrow^{n+1} w$ in H. Then, $S \Rightarrow^n v \Rightarrow w$, where $v \in V^*$, and there exists $r \in P$ such that $v \Rightarrow w\,[r]$. By the induction hypothesis, $\langle S'|s|\varepsilon|\varepsilon|\varepsilon\rangle \Rightarrow^* v'$, where $\gamma(v') = v$, in G'. Next, we consider the following five forms of r according to the construction of G'.



$\langle S'|s|\varepsilon|\varepsilon|\varepsilon|\varepsilon\rangle$
$\langle S|s|\varepsilon|\varepsilon|p|f\rangle$
$s1 \vdash s$
a
$\langle S|s|\varepsilon|\varepsilon|p|f\rangle$
b
$s2_1 \vdash s_q$
$\langle\langle 2\rangle|s_q|p|\varepsilon|\varepsilon|\varepsilon\rangle$
$\langle B|p|\varepsilon|\varepsilon|\varepsilon|f\rangle$
$s_q 2_2 \vdash q$
$p5 \vdash p$
$\langle A|q|p|\varepsilon|\varepsilon|\varepsilon\rangle$
e
$\langle B|p|\varepsilon|\varepsilon|\varepsilon|f\rangle$
f
$q3 \vdash q$
$p6_1 \vdash p_f$
c
$\langle A|q|p|\varepsilon|\varepsilon|\varepsilon\rangle$
d
$\langle\langle 4\rangle|p_f|\varepsilon|\varepsilon|\varepsilon|f\rangle$
$q4_1 \vdash q_p$
$p_f 6_2 \vdash f$
$\langle\langle 6\rangle|q_p|p|\varepsilon|\varepsilon|\varepsilon\rangle$
ε
$q_p 4_2 \vdash p$
ε

Figure 6.2.1: ${}_{G'}\triangle(\langle S'|s|\varepsilon|\varepsilon|\varepsilon|\varepsilon\rangle \Rightarrow^* acdefb\,[\dot{0}\dot{1}\dot{2}\dot{3}\dot{4}\dot{5}\dot{6}\dot{7}\dot{8}\dot{9}])$

(1) Let $r\colon A \to u_1Bu_2 \in P$, for some $A, B \in N$, $u_1, u_2 \in T^*$, and $v \Rightarrow w\,[r]$ is a path-preserving derivation step or a path-changing derivation step into a node with some sibling. By the construction of G', there exists a rule $\langle A|q|x|y|z|f\rangle \to u_1\langle B|p|x|y|z|f\rangle u_2$ in P', where $qr \vdash p \in R$, $x, y, z \in (Q \cup \{\varepsilon\})^k$, and $f \in F \cup \{\varepsilon\}$. Without any loss of generality, suppose q, x, y, z, f are correct. Then, there exists a derivation

$$v' \Rightarrow w' \ [\langle A|q|x|y|z|f\rangle \to u_1\langle B|p|x|y|z|f\rangle u_2]$$

in G', where $\gamma(w') = w$.

(2) Let $r\colon A \to u_1Bu_2 \in P$, for some $A, B \in N$, $u_1, u_2 \in T^*$, and $v \Rightarrow w\,[r]$ is a path-changing derivation step into a node without siblings. By the construction of G', there exists a rule $\langle A|g|gx|qy|z|f\rangle \to u_1\langle B|p|x|y|z|f\rangle u_2$ in P', where $g \in Q$, $qr \vdash p \in R$, $x, y, z \in (Q \cup \{\varepsilon\})^k$, and $f \in F \cup \{\varepsilon\}$. Without any loss of generality, suppose g, q, x, y, z, f are correct. Then, there exists a derivation

$$v' \Rightarrow w' \ [\langle A|g|gx|qy|z|f\rangle \to u_1\langle B|p|x|y|z|f\rangle u_2]$$

in G', where $\gamma(w') = w$.

(3) Let $r\colon A \to x \in P$, for some $A \in N$, $x \in T^*$, and $\mathrm{alph}(w) \cap N \neq \emptyset$. By the construction of G', there exists a rule $\langle A|q|p|\varepsilon|\varepsilon|\varepsilon\rangle \to x$ in P', where $qr \vdash p \in R$. Without any loss of generality, suppose q is correct. Then, there exists a derivation

$$v' \Rightarrow w' \ [\langle A|q|p|\varepsilon|\varepsilon|\varepsilon\rangle \to x]$$

in G', where $\gamma(w') = w$.

(4) Let $r\colon A \to x \in P$, for some $A \in N$, $x \in T^*$, and $w \in T^*$. By the construction of G', there exists a rule $\langle A|q|\varepsilon|\varepsilon|\varepsilon|f\rangle \to x$ in P', where $qr \vdash f \in R$, $f \in F$. Without any loss of generality, suppose q is correct. Then, there exists a derivation

$$v' \Rightarrow w' \ [\langle A|q|\varepsilon|\varepsilon|\varepsilon|f\rangle \to x]$$

in G', where $\gamma(w') = w$.

(5) Let $r\colon A \to BC \in P$, for some $A, B, C \in N$. By the construction of G', there exists a rule $\langle A|q|x_1|y_1|z_1|f_1\rangle \to \langle B|p|x_2|y_2|z_2|f_2\rangle\langle C|g|x_3|y_3|z_3|f_3\rangle$ in P', where $qr \vdash p \in R$, $x_i,y_i,z_i \in (Q \cup \{\varepsilon\})^k$, $f_i \in F \cup \{\varepsilon\}$, and without any loss of generality, suppose q,g,x_i,y_i,z_i,f_i are correct, for $1 \le i \le 3$. Then, there exists a derivation

$$v' \Rightarrow w' \ [\langle A|q|x_1|y_1|z_1|f_1\rangle \to \langle B|p|x_2|y_2|z_2|f_2\rangle\langle C|g|x_3|y_3|z_3|f_3\rangle]$$

in G', where $\gamma(w') = w$.

We covered all possible forms of p, so the claim holds. □

Let us remark that assumption of correctness of nonterminals of G' results from the fact that the rules cover all possibilities—that is there is always a proper rule to be used.

Claim 7. Consider any $w \in T^*$, where $w \notin \mathrm{L}(H)$. Then, $w \notin \mathrm{L}(G')$.

Proof. We prove this by contradiction.

Assumption. Suppose there exists $w \in T^*$, where $w \notin \mathrm{L}(H)$ and $w \in \mathrm{L}(G')$.

1. First, suppose $w \notin \mathrm{L}(G)$. That is, there exists a derivation

$$\langle S'|s|\varepsilon|\varepsilon|\varepsilon|\varepsilon\rangle \Rightarrow \langle S|s|\varepsilon|\varepsilon|x|f\rangle \Rightarrow^* u \Rightarrow v\,[A \to X] \Rightarrow^* w,$$

in G', where $\gamma(u) \not\Rightarrow \gamma(v)$ in G, for some $u, v \in V'^*$, $\langle S|s|\varepsilon|\varepsilon|x|f\rangle \in N'$, and $A \to X \in P'$. Then, $\gamma(A) \to \gamma(X) \notin P$. However, since by the construction of G' every non-initial rule $A \to X \in P'$ is introduced according to some $\gamma(A) \to \gamma(X) \in P$, this is a contradiction.

2. Second, suppose $w \in \mathrm{L}(G)$, however, for every derivation $S \Rightarrow^* w\,[d]$ in G, $d \notin C$. In the terms of M, there is no derivation $sd \vdash^* \bar{q}$, for any $\bar{q} \in Q$, or $sd \vdash \bar{q}$ and $\bar{q} \notin F$. Consider a derivation

$$\langle S'|s|\varepsilon|\varepsilon|\varepsilon|\varepsilon\rangle \Rightarrow \langle S|s|\varepsilon|\varepsilon|\bar{x}|\bar{f}\rangle \Rightarrow^* w,$$

in G', for some $\langle S|s|\varepsilon|\varepsilon|\bar{x}|\bar{f}\rangle \in N'$, and a corresponding derivation $S \Rightarrow^* w\,[d]$ in G, for some $d \in R^*$.

   (a) Suppose there is no derivation $sd \vdash^* \bar{q}$, for any $\bar{q} \in Q$. Then, there exist $u, v \in V'^*$ and $\langle A|q|x|y|z|f\rangle \to X \in P'$, where

$$\langle S'|s|\varepsilon|\varepsilon|\varepsilon|\varepsilon\rangle \Rightarrow \langle S|s|\varepsilon|\varepsilon|\bar{x}|\bar{f}\rangle \Rightarrow^* u \Rightarrow v\,[\langle A|q|x|y|z|f\rangle \to X] \Rightarrow^* w,$$

   in G', and a corresponding derivation

$$S \Rightarrow^* \gamma(u)\,[d_1] \Rightarrow \gamma(v)\,[r] \Rightarrow^* w\,[d_2]$$

   in G, where $d_1rd_2 = d$, $r\colon A \to \gamma(X) \in P$, $sd_1 \vdash^* q$, and there is no derivation $sd_1r \vdash^* q'$, for any $q, q' \in Q$.

i. Suppose $\gamma(u) \Rightarrow \gamma(v)\,[r]$ is a path-preserving derivation step. Then,

$$\langle A|q|x|y|z|f\rangle \to X$$

is from (II) or (IV) through (VI) depending on X. By the construction of G', the rule $\langle A|q|x|y|z|f\rangle \to X$ is introduced according to a transition $qr \vdash p \in R$, for some $p \in Q$. Therefore, however, $sd_1r \vdash^* p$ in M, which is a contradiction.

ii. Suppose $\gamma(u) \Rightarrow \gamma(v)\,[r]$ is a path-changing derivation step. The states in which the path-changing derivation steps are always to be performed—represented by the string $\bar{x}$—are nondeterministically generated by the initial derivation step

$$\langle S'|s|\varepsilon|\varepsilon|\varepsilon\rangle \Rightarrow \langle S|s|\varepsilon|\varepsilon|\bar{x}|\bar{f}\rangle$$

in the fifth component of $\langle S|s|\varepsilon|\varepsilon|\bar{x}|\bar{f}\rangle$. Therefore, $q \in \mathrm{alph}(\bar{x})$. However, before the path-change can be simulated by G', the state must get to the third and fourth component of some nonterminals which can be done by the rule (VI) only. Then, by some rule

$$\langle A_1|q_1|x_1|y_1|z_1|f_1\rangle \to \langle A_2|q_2|x_2|y_2|z_2|f_2\rangle\langle A_3|q_3|x_3|y_3|z_3|f_3\rangle \in P',$$

where $\#_q(z_1) \geq \#_q(z_2) + \#_q(z_3) + 1$,

$$\begin{aligned}\langle S|s|\varepsilon|\varepsilon|\bar{x}|\bar{f}\rangle &\Rightarrow^* u_1\langle A_1|q_1|x_1|y_1|z_1|f_1\rangle u_2\\ &\Rightarrow u_1\langle A_2|q_2|x_2|y_2|z_2|f_2\rangle\langle A_3|q_3|x_3|y_3|z_3|f_3\rangle u_2,\end{aligned}$$

for some $u_1, u_2 \in V'^*$. Therefore, $q \in \mathrm{alph}(x_2)$ or $q \in \mathrm{alph}(x_3)$ and since these two cases are symmetric, without any loss of generality, let us consider only $q \in \mathrm{alph}(x_2)$. Then, also $q_3 = q$ and

$$\langle A_3|q_3|x_3|y_3|z_3|f_3\rangle = \langle A_3|q|x_3|y_3|z_3|f_3\rangle$$

or $q \in \mathrm{alph}(y_3)$ and to once get rid of it

$$\langle A_3|q_3|x_3|y_3|z_3|f_3\rangle \Rightarrow^* w_1\langle A|g|gx|qy|z|f\rangle w_2$$

for some $w_1, w_2 \in V'^*$. Either $\langle A_3|q|x_3|y_3|z_3|f_3\rangle$ or $\langle A|g|gx|qy|z|f\rangle$ represents the target of the path-change later denoted by Z. Additionally, G' must once get rid of q in x_2, otherwise, the derivation is not successful. Hence,

$$\langle A_2|q_2|x_2|y_2|z_2|f_2\rangle \Rightarrow^* v_1Yv_2,$$

where $Y = \langle A_4|q|qx_4|y_4|z_4|f_4\rangle$ which consequently allows an application of a rule (III) erasing q, for some $v_1, v_2 \in V'^*$ and $\langle A_4|q|qx_4|y_4|z_4|f_4\rangle \in N'$, or $Y \in T^*$ and

$$\begin{aligned}\langle A_2|q_2|x_2|y_2|z_2|f_2\rangle &\Rightarrow^* v_1\langle A'|q'|q|\varepsilon|\varepsilon|\varepsilon\rangle v_2\,[\langle A'|q'|q|\varepsilon|\varepsilon|\varepsilon\rangle \to Y]\\ &\Rightarrow v_1Yv_2,\end{aligned}$$

for some $\langle A'|q'|q|\varepsilon|\varepsilon|\varepsilon\rangle \to Y \in P'$. Combining the previous observations and statements, we get

$$\langle S'|s|\varepsilon|\varepsilon|\varepsilon|\varepsilon\rangle \Rightarrow^* u_1v_1Yv_2w_1Zw_2u_2.$$

Since G' is a context-free grammar, without any loss of generality, we can suppose that the derivation follows M—that is, $sd_1 \vdash q \in M$. Then, the derivation

$$u_1v_1Yv_2w_1Zw_2u_2 \Rightarrow u_1v_1Yv_2w_1Xw_2u_2$$

represents a path-changing derivation step changing the path of the derivation from Y to X. By the construction of G', the rule $\langle A|q|x|y|z|f\rangle \to X$ or $\langle A|g|gx|qy|z|f\rangle \to X$ by which the last derivation step is performed is introduced according to a transition $qr \vdash p \in R$, for some $p \in Q$. Therefore, however, $sd_1r \vdash^* p$ in M, which is a contradiction.

(b) Suppose $sd \vdash^* \bar{q}$, where $\bar{q} \in Q - F$. In every successful derivation of G', there is precisely one rule (V) applied—G' must once get rid of $\bar{f}$ generated by the initial derivation step—which represents final accepting transition of M. Since G' is a context free grammar, without any loss of generality, we can consider an application of such rule is always performed at the end of any successful derivation; this is also consistent with M and, thus, C. Then, however, $\bar{q} \in F$, which is a contradiction.

Since the assumption always results in contradiction, it is incorrect. □

By Claim 6, if $S \Rightarrow^* w$ in H, then $\langle S'|s|\varepsilon|\varepsilon|\varepsilon|\varepsilon\rangle \Rightarrow^* w'$ in G', where $\gamma(w') = w$. If $S \Rightarrow^* w$ in H and $w \in T^*$, then $w \in \mathrm{L}(H)$. Since $\gamma(w') = w' = w$, for $w \in T^*$, $w' \in \mathrm{L}(G')$. By Claim 7, $\mathrm{L}(G') - \mathrm{L}(H) = \emptyset$. Therefore, $\mathrm{L}(H) = \mathrm{L}(G')$. By Claim 5 $\mathrm{L}(\bar{H}) = \mathrm{L}(H)$ and Theorem 6.2.1 holds. □

Corollary 6.2.1. *Let L be a context-free language of an infinite index. Then, there exists no k-restricted regular-controlled grammar H such that $\mathrm{L}(H) = L$, for any $1 \leq k < \infty$.*

Corollary 6.2.2. *If there is a constant $k \geq 0$ and a propagating regular-controlled grammar H in binary form such that, for every $w \in L(H)$, there exists a derivation of w in H with at most k path-changing derivation steps, then $L(H)$ is a context-free language, and moreover, it is of index $k + 1$.*

We introduced the binary form of regular-controlled grammars to simplify the proof of Theorem 6.2.1. Notice, however, it can be generalized for all k-restricted RCGs. A proof of the following theorem is, thus, left to the reader.

Theorem 6.2.2. *If there is a constant $k \geq 0$ and a regular-controlled grammar H such that, for every $w \in L(H)$, there exists a derivation of w in H with at most k path-changing derivation steps, then $L(H)$ is a context-free language, and moreover, it is of index $k + 1$.*

The control mechanism of regular-controlled grammars influences the order in which the core grammars apply their rules. However, the notion of path-change as well as the given restrictions are independent of this control mechanism and are related only to the core grammars and their derivation trees. Therefore, we can state the achieved result in a more general context.

Corollary 6.2.3. *If there is a constant $k \geq 0$ and a (propagating) matrix grammar H such that, for every $w \in L(H)$, there exists a derivation of w in H with at most k path-changing derivation steps, then $L(H)$ is a context-free language, and moreover, it is of index $k + 1$.*

6.3 Scattered Context Grammars

Parallelism as a computational phenomenon stands indisputably as a crucial area of interest for theoretical computer scientists for long decades. In this section we focus on first grammatical model working in parallel, scattered context grammars (see Section 2.2). Originally, scattered context grammars were defined in [20], later in [56] their generalized versions with erasing rules (see also [34]) were introduced. They were widely studied in numerous publications (see for example [13, 16, 17, 29–33, 36, 37, 39, 41, 44, 52, 53, 55]). For a detailed inside into scattered context grammars consult [40].

The general versions of scattered context grammars characterize the family of recursively enumerable languages and are, thus, computationally complete, while their propagating versions characterize the family of context sensitive languages. In essence, an SCG is a context-free grammar which, however, possibly apply several context-free rules in parallel. In this way the grammar in fact introduces context-dependencies between simultaneously rewritten symbols. Since they are equally powerful (see Theorem 2.2.10), let us consider only SCGs in the binary form. From the derivation-tree point of view, there occur pairs of context-dependent nodes. Let $k \geq 0$ be a constant. In what follows, we show that if these context-dependent pairs of nodes are clustered into mutually context-independent (k or less)-tuples, the generated language is context-free. Let us emphasize that we do not limit the total number of context dependencies at all. Moreover, this result can be of some use in practice, since we can obtain a positive proof of context-freeness based on it.

Theorem 6.3.1. *A language L is context-free iff there is a constant $k \geq 0$ and a scattered context grammar G in the binary form such that $L = \mathrm{L}(G)$ and for every $x \in \mathrm{L}(G)$, there is a tree $t = {}_G\triangle_x \in {}_G\blacktriangle$ for which there exists a division t' such that in every subgraph of t' there are k or fewer pairs of context-dependent nodes and every pair of nodes from two different subgraphs is context-independent.*

We divide the proof into only if and if part.

Proof. Only If. Let L be a context-free language. Then, there exists a context-free grammar G, where $\mathrm{L}(G) = L$. However, G is in fact also scattered context grammar without non-context-free rules. Without any loss of generality, suppose G is in the Chomsky Normal form (see Definition 2.2.16); then, G also satisfies the binary form. For any $x \in \mathrm{L}(G)$ there is a tree $t = {}_G\triangle_x \in {}_G\blacktriangle$ and a division $t' = t$; the 1-division of t. Since G has no non-context-free rules, there are no context-dependent nodes in t' so $k = 0$ which completes the only if part of the proof.

If. Construction. Consider any $k \geq 0$. Let $G = (V, T, P, S)$ be an SCG which satisfies restrictions from Theorem 6.3.1 such that $\mathrm{L}(G) = L$. Recall $N = V - T$. Let $P_{cs} \subseteq P$ denote the set of all non-context-free rules of G. Set

$$N' = \{A_{x|y|z} \mid A \in N,\ x, y, z \in (P_{cs} \cup \{\varepsilon\})^k\}.$$

Construct a grammar $G' = (V', T, P', S_{\varepsilon|\varepsilon|\varepsilon})$, where $V' = N' \cup T$. Set $P' = \emptyset$. Construct P' by performing (I) through (V) given next.

(I) For all $A \in N$ and $x \in (P_{cs} \cup \{\varepsilon\})^k$, add $A_{\varepsilon|\varepsilon|\varepsilon} \to A_{x|\varepsilon|\varepsilon}$ to P';

(II) For all $(A) \to (B) \in P$, $A, B \in N$, and $x, y, z \in (P_{cs} \cup \{\varepsilon\})^k$, add $A_{x|y|z} \to B_{x|y|z}$ to P';

(III) for all $(A) \to (BC) \in P$, where $A, B, C \in N$, $x, y, z \in (P_{cs} \cup \{\varepsilon\})^k$, $x = x_1x_2x_3$, $y = \text{shuffle}(y_1, y_2)$, and $z = \text{shuffle}(z_1, z_2)$, add

$$A_{x|y|z} \to B_{x_1|y_1x_3|z_1} C_{x_2|y_2|z_2x_3} \text{ to } P';$$

(IV) for all $p\colon (A, B) \to (C, D) \in P$, where $A, B, C, D \in N$, and $x, y, z, x', y', z' \in (P_{cs} \cup \{\varepsilon\})^k$, add $A_{x|py|z} \to C_{x|y|z}$ and $B_{x'|y'|pz'} \to D_{x'|y'|z'}$ to P';

(V) for all $(A) \to (a) \in P$, $A \in N$, $a \in (T \cup \{\varepsilon\})$, add $A_{\varepsilon|\varepsilon|\varepsilon} \to a$ to P'.

For every rule $X \to Y \in P'$, introduce a unique label p such that $p\colon X \to Y$.

Basic idea. From the derivation-tree-based restrictions we know that all the context dependencies are in fact divided into at maximum k-tuples each of which is located inside a specific subgraph of the derivation tree without any relation to the outer nodes. Then, G can store them in nonterminals. Since any connected subgraph of tree is also a tree, it has a root node which is obviously context-independent. During the derivation while introducing nonterminal corresponding to the root node of a new subgraph, G' nondeterministically decides about all the context dependencies inside the subgraph with a rule (I), distributes them by rules (II) and (III), satisfies them by rules (IV), and finally generates a terminal string by rules (V).

Since P' contains no non-context-free rule, G' is context-free. First, we establish several preliminary claims. Claims 8 through 12 given next prove that the derivations in G' coincides with the subgraph-division-based structure of G.

Claim 8. Let $S_{\varepsilon|\varepsilon|\varepsilon} \Rightarrow^*_{G'} w$, where $w \in T^*$. Then, $S_{\varepsilon|\varepsilon|\varepsilon} \Rightarrow^*_{G'} v\ [\varrho] \Rightarrow^*_{G'} w\ [\varphi]$, where ϱ contains no rules (V) and φ contains only rules (V).

Proof. Rules (V) generate only terminal symbols. Since G' is context-free grammar, applications of these rules may be postponed to the very end of every derivation, without any loss of generality, to satisfy the claim. □

Claim 9. Let $S_{\varepsilon|\varepsilon|\varepsilon} \Rightarrow^*_{G'} w \Rightarrow^*_{G'} w'$, where $w = uA_{x|y|z}v$, for some $u, v \in V'^*$, $A_{x|y|z} \in N'$, $x, y, z \in (P_{cs} \cup \{\varepsilon\})^k$, and $w' \in T^*$. Then, $A_{x|y|z} \Rightarrow^*_{G'} a_1a_2 \dots a_n$, for some $n \geq 0$, where $a_i = A_{i\varepsilon|\varepsilon|\varepsilon}$, for some $A_i \in N$, $1 \leq i \leq n$.

Proof. Suppose that G' satisfies Claim 8. Then,

$$S_{\varepsilon|\varepsilon|\varepsilon} \Rightarrow^*_{G'} w \Rightarrow^*_{G'} w'' \Rightarrow^*_{G'} w'\ [\varphi],$$

where $w, w'' \in N'^*$ and φ contains only rules (V). Since these rules are of the form $A_{\varepsilon|\varepsilon|\varepsilon} \to a$, $w'' = A_{1\varepsilon|\varepsilon|\varepsilon}A_{2\varepsilon|\varepsilon|\varepsilon} \dots A_{k\varepsilon|\varepsilon|\varepsilon}$, $A_i \in N$, $1 \leq i \leq k$, for some $k \geq 0$, and the claim holds. □

Claim 10. Let

$$\begin{aligned} S_{\varepsilon|\varepsilon|\varepsilon} &\Rightarrow^*_{G'} u_1 \Rightarrow_{G'} v_1 \quad [r_1] \\ &\Rightarrow^*_{G'} u_2 \Rightarrow_{G'} v_2 \quad [r_2] \\ &\quad\vdots \\ &\Rightarrow^*_{G'} u_n \Rightarrow_{G'} v_n \quad [r_n] \Rightarrow^*_{G'} w, \end{aligned}$$

where $w \in T^*$, $u_i, v_i \in V'^*$, $1 \leq i \leq n$, and $r_1, r_2, \dots, r_n$ are all the rules (I) applied in the derivation, for some $n \geq 0$. Then, there exists a derivation

$$\begin{aligned} S_{\varepsilon|\varepsilon|\varepsilon} \Rightarrow^*_{G'} \quad & u'_1 \Rightarrow_{G'} v'_1 \quad [r_1] \\ \Rightarrow^*_{G'} \quad & u'_2 \Rightarrow_{G'} v'_2 \quad [r_2] \\ & \vdots \\ \Rightarrow^*_{G'} \quad & u'_n \Rightarrow_{G'} v'_n \quad [r_n] \Rightarrow^*_{G'} w, \end{aligned}$$

where $u_i = a_1 a_2 \dots a_{k_i}$, for some $k_i \geq 0$, and $a_j = A_{j_{\varepsilon|\varepsilon|\varepsilon}}$, for some $A_j \in N$, $1 \leq j \leq k_i$, $1 \leq i \leq n$.

Proof. We establish the proof by induction on $n \geq 0$.

Basis. Let $n = 0$. Then, the basis holds trivially.

Induction Hypothesis. Suppose that there exists $m \geq 0$ such that Claim 10 holds for all n with $0 \leq n \leq m$.

Induction Step. Consider a derivation

$$\begin{aligned} S_{\varepsilon|\varepsilon|\varepsilon} \Rightarrow^*_{G'} \quad & u_1 \Rightarrow_{G'} v_1 \quad & [r_1] \\ \Rightarrow^*_{G'} \quad & u_2 \Rightarrow_{G'} v_2 \quad & [r_2] \\ & \vdots \\ \Rightarrow^*_{G'} \quad & u_m \Rightarrow_{G'} v_m \quad & [r_m] \\ \Rightarrow^*_{G'} \quad & u_{m+1} \Rightarrow_{G'} v_{m+1} \quad & [r_{m+1}] \Rightarrow^*_{G'} w, \end{aligned}$$

where $w \in T^*$, $u_i, v_i \in V'^*$, $1 \leq i \leq m+1$, and $r_1, r_2, \dots, r_m, r_{m+1}$ are all the rules (I) applied in the derivation. Since $r_{m+1}\colon A_{\varepsilon|\varepsilon|\varepsilon} \rightarrow A_{x|\varepsilon|\varepsilon}$, for some $A \in N$ and $x \in (P_{cs} \cup \{\varepsilon\})^k$, $u_{m+1} = \alpha A_{\varepsilon|\varepsilon|\varepsilon} \beta$, for some $\alpha, \beta \in V'^*$. Without any loss of generality, suppose the derivation satisfies Claim 8. Then, $\alpha, \beta \in N'^*$ and by Claim 9,

$$\begin{aligned} \alpha &\Rightarrow^*_{G'} a_1 a_2 \dots a_k \\ \beta &\Rightarrow^*_{G'} b_1 b_2 \dots b_l \end{aligned}$$

for some $k, l \geq 0$, where $a_i = A_{i_{\varepsilon|\varepsilon|\varepsilon}}$, for some $A_i \in N$, $1 \leq i \leq k$, and $b_j = B_{j_{\varepsilon|\varepsilon|\varepsilon}}$, for some $B_j \in N$, $1 \leq j \leq l$. Therefore,

$$S_{\varepsilon|\varepsilon|\varepsilon} \Rightarrow^*_{G'} \alpha A_{\varepsilon|\varepsilon|\varepsilon} \beta \Rightarrow^*_{G'} a_1 a_2 \dots a_k A_{\varepsilon|\varepsilon|\varepsilon} b_1 b_2 \dots b_l$$

which satisfies the claim and, thus, completes the proof. □

Claim 11. Let

$$S_{\varepsilon|\varepsilon|\varepsilon} \Rightarrow^*_{G'} u A_{\varepsilon|\varepsilon|\varepsilon} v \Rightarrow_{G'} u A_{x|\varepsilon|\varepsilon} v \; [r] \Rightarrow^*_{G'} w,$$

where $w \in T^*$, $u, v \in V'^*$, and $r\colon A_{\varepsilon|\varepsilon|\varepsilon} \rightarrow A_{x|\varepsilon|\varepsilon}$ is a rule (I), for some $A \in N$, $x \in (P_{cs} \cup \{\varepsilon\})^k$. Then,

$$S_{\varepsilon|\varepsilon|\varepsilon} \Rightarrow^*_{G'} u A_{\varepsilon|\varepsilon|\varepsilon} v \Rightarrow_{G'} u A_{x|\varepsilon|\varepsilon} v \; [r] \Rightarrow^*_{G'} u a_1 a_2 \dots a_k v \; [\varrho] \Rightarrow^*_{G'} w,$$

where $a_i = A_{i_{\varepsilon|\varepsilon|\varepsilon}}$, for some $A_i \in N$, $1 \leq i \leq k$.

Proof. By Claim 9, $A_{x|\varepsilon|\varepsilon} \Rightarrow^*_{G'} a_1 a_2 \dots a_k$, where $a_i = A_{i_{\varepsilon|\varepsilon|\varepsilon}}$, for some $A_i \in N$, $1 \leq i \leq k$, and since G' is context-free grammar, without any loss of generality, we can always reorder the rules in the derivation to satisfy Claim 11. □

Claim 12. Every terminal derivation of G' is performed as

$$\begin{array}{lll} S_{\varepsilon|\varepsilon|\varepsilon} \Rightarrow^*_{G'} & A_{1,1_{\varepsilon|\varepsilon|\varepsilon}} A_{1,2_{\varepsilon|\varepsilon|\varepsilon}} \cdots A_{1,i_1{}_{\varepsilon|\varepsilon|\varepsilon}} \cdots A_{1,n_1{}_{\varepsilon|\varepsilon|\varepsilon}} & \\ \Rightarrow_{G'} & A_{1,1_{\varepsilon|\varepsilon|\varepsilon}} A_{1,2_{\varepsilon|\varepsilon|\varepsilon}} \cdots A_{1,i_1{}_{x_1|\varepsilon|\varepsilon}} \cdots A_{1,n_1{}_{\varepsilon|\varepsilon|\varepsilon}} & [r_1] \\ & & \\ \Rightarrow^*_{G'} & A_{2,1_{\varepsilon|\varepsilon|\varepsilon}} A_{2,2_{\varepsilon|\varepsilon|\varepsilon}} \cdots A_{2,i_2{}_{\varepsilon|\varepsilon|\varepsilon}} \cdots A_{2,n_2{}_{\varepsilon|\varepsilon|\varepsilon}} & \\ \Rightarrow_{G'} & A_{2,1_{\varepsilon|\varepsilon|\varepsilon}} A_{2,2_{\varepsilon|\varepsilon|\varepsilon}} \cdots A_{2,i_2{}_{x_2|\varepsilon|\varepsilon}} \cdots A_{2,n_2{}_{\varepsilon|\varepsilon|\varepsilon}} & [r_2] \\ & \vdots & \\ \Rightarrow^*_{G'} & A_{m,1_{\varepsilon|\varepsilon|\varepsilon}} A_{m,2_{\varepsilon|\varepsilon|\varepsilon}} \cdots A_{m,i_m{}_{\varepsilon|\varepsilon|\varepsilon}} \cdots A_{m,n_m{}_{\varepsilon|\varepsilon|\varepsilon}} & \\ \Rightarrow_{G'} & A_{m,1_{\varepsilon|\varepsilon|\varepsilon}} A_{m,2_{\varepsilon|\varepsilon|\varepsilon}} \cdots A_{m,i_m{}_{x_m|\varepsilon|\varepsilon}} \cdots A_{m,n_m{}_{\varepsilon|\varepsilon|\varepsilon}} & [r_m] \\ & & \\ \Rightarrow^*_{G'} & A_{m+1,1_{\varepsilon|\varepsilon|\varepsilon}} A_{m+1,2_{\varepsilon|\varepsilon|\varepsilon}} \cdots A_{m+1,n_{m+1}{}_{\varepsilon|\varepsilon|\varepsilon}} & \\ \Rightarrow^*_{G'} & w\ [\varrho], & \end{array}$$

where $w \in T^*$, $A_{i,j_i} \in N$, $x_i \in (P_{cs} \cup \{\varepsilon\})^k$, $1 \le i \le m$, $1 \le j_i \le n_i$, $r_1, r_2, \dots, r_m$ are all the rules (I) applied in the derivation, for some $m, n_i \ge 0$, ϱ contains only the rules (V), there are no other rules (V) applied, and every derivation

$$\begin{array}{ll} & A_{j,1_{\varepsilon|\varepsilon|\varepsilon}} A_{j,2_{\varepsilon|\varepsilon|\varepsilon}} \cdots A_{j,i_j{}_{x_j|\varepsilon|\varepsilon}} \cdots A_{j,n_j{}_{\varepsilon|\varepsilon|\varepsilon}} \\ \Rightarrow^*_{G'} & A_{j+1,1_{\varepsilon|\varepsilon|\varepsilon}} A_{j+1,2_{\varepsilon|\varepsilon|\varepsilon}} \cdots \quad \cdots A_{j+1,n_{j+1}{}_{\varepsilon|\varepsilon|\varepsilon}} \quad [\varphi] \end{array}$$

where φ contains no rule (I), can be expressed as $uA_{j,i_j{}_{x_j|\varepsilon|\varepsilon}}v \Rightarrow^*_{G'} uxv\ [\varphi]$, for some $u, v, x \in V'^*$, $0 \le j \le m$.

Proof. The claim follows from Claims 8 through 11. □

The following Claims 13 through 17 help to prove that G' simulates non-context-free rules of G correctly.

Claim 13. Consider a derivation satisfying Claim 12,

$$\begin{array}{lll} S_{\varepsilon|\varepsilon|\varepsilon} \Rightarrow^*_{G'} & uA_{\varepsilon|\varepsilon|\varepsilon}v & \\ \Rightarrow_{G'} & uA_{p_1p_2\cdots p_n|\varepsilon|\varepsilon}v & [r] \\ \Rightarrow^*_{G'} & uB_{1_{\varepsilon|\varepsilon|\varepsilon}}B_{2_{\varepsilon|\varepsilon|\varepsilon}} \cdots B_{m_{\varepsilon|\varepsilon|\varepsilon}}v & [\varrho] \\ \Rightarrow^*_{G'} & w & \end{array}$$

where $w \in T^*$, $u, v \in V'^*$, $A, B_i \in N$, $p_j \in P_{cs}$, $0 \le i \le m$, $0 \le j \le n$, for some $m \ge 0$, $n \ge 1$, $r \in P'$ is a rule (I), and $\varrho \in P'^*$ contains no rule (I). Then, for every p_j,

$$uA_{p_1p_2\cdots p_j\cdots p_n|\varepsilon|\varepsilon}v \quad \Rightarrow^*_{G'} uu_1B_{x|y_1p_jy_2|z}u_2v \quad [\varphi]$$

where $\varrho = \varphi\sigma$, for some $\sigma \in P'^*$, $u_1, u_2 \in V'^*$, $B \in N$, and x, y_1, y_2, $z \in (P_{cs} \cup \{\varepsilon\})^k$.

Proof. Since $n \ge 1$, $p_1p_2 \cdots p_n \ne \varepsilon$ in $A_{p_1p_2\cdots p_n|\varepsilon|\varepsilon}$.

$$A_{p_1p_2\cdots p_n|\varepsilon|\varepsilon} \Rightarrow^*_{G'} B_{1_{\varepsilon|\varepsilon|\varepsilon}}B_{2_{\varepsilon|\varepsilon|\varepsilon}} \cdots B_{m_{\varepsilon|\varepsilon|\varepsilon}}\ [\varrho],$$

so G' must once get rid of $p_1p_2 \cdots p_n$. Observe the rules of G'. This obviously cannot be done by any rule (I), (II), and (V). The rules (IV) are of the form $A_{x|y|z} \to B_{x'|y'|z'}$, where $|xyz| = |x'y'z'| + 1$, however, $x = x'$. Thus, their applicability depends on the rules (III) since they are of the form $A_{x_1x_2x_3,y,z} \rightarrow B_{x_1,y'x_3,z'}C_{x_2,y'',z''x_3}$, where $|x_1x_2x_3| \geq |x_1x_2|$, $|y'x_3| \geq |y'|$, and $|z''x_3| \geq |z''|$, for some $A_{x_1x_2x_3,y,z}, B_{x_1,y'x_3,z'}, C_{x'',y'',z''x_3} \in N'^*$. So, for every p_j in $A_{p_1p_2\cdots p_n|\varepsilon|\varepsilon}$, $0 \leq j \leq n$, there is $p'_j \in P'$ such that

$$p'_j\colon X_{x_1x_2x_3|y|z} \rightarrow Y_{x_1|y'x_3|z'}Z_{x_2|y''|z''x_3},$$

where $\varrho = \varrho_1 p'_j \varrho_2$, $x_3 = x_a p_j x_b$, for some $X_{x_1x_2x_3|y|z}, Y_{x_1|y'x_3|z'}, Z_{x_2|y''|z''x_3} \in N'$. Then,

$$\begin{array}{lll}
uA_{p_1p_2\cdots p_j\cdots p_n|\varepsilon|\varepsilon}v & \Rightarrow^*_{G'} uu'X_{x_1x_2x_3|y|z}u''v & [\varrho_1] \\
& \Rightarrow_{G'} uu'Y_{x_1|y'x_ap_jx_b|z'}Z_{x_2|y''|z''x_3}u''v & [p'_j] \\
& \Rightarrow^*_{G'} B_{1\varepsilon|\varepsilon|\varepsilon}B_{2\varepsilon|\varepsilon|\varepsilon}\cdots B_{m\varepsilon|\varepsilon|\varepsilon} & [\varrho_2]
\end{array}$$

which completes the proof. □

Claim 14. Let $S_{\varepsilon|\varepsilon|\varepsilon} \Rightarrow^*_{G'} uA_{x|y_1py_2|z}v \Rightarrow^*_{G'} w$, where $w \in T^*$, $u, v \in V'^*$, $A_{x|y_1py_2|z} \in N'$, and $p \in P_{cs}$. Then,

$$S_{\varepsilon|\varepsilon|\varepsilon} \Rightarrow^*_{G'} uA_{x|y_1py_2|z}v'B_{x'|y|z_1pz_2}w' \Rightarrow^*_{G'} w,$$

where $B_{x'|y|z_1pz_2} \in N'$, $v', w' \in V'^*$.

Proof. Consider a derivation $S_{\varepsilon|\varepsilon|\varepsilon} \Rightarrow^*_{G'} uA_{x|y_1py_2|z}v \Rightarrow^*_{G'} w$, where $u, v \in V'^*$, $A_{x|y_1py_2|z} \in N'$, $p \in P_{cs}$, and $w \in T^*$. Then, by the construction of P',

$$\begin{array}{lll}
S_{\varepsilon|\varepsilon|\varepsilon} & \Rightarrow^*_{G'} u'X_{x_1x_2x_3px_4|y'|z'}v' & \\
& \Rightarrow_{G'} u'Y_{x_1|y'_1x_3px_4|z'_1}Z_{x_2|y'_2|z'_2x_3px_4}v' & [r] \\
& \Rightarrow^*_{G'} uA_{x|y_1py_2|z}v & \\
& \Rightarrow^*_{G'} w &
\end{array}$$

for some $u', v' \in V'^*$, $X_{x_1x_2x_3px_4|y'|z'}, Y_{x_1|y'_1x_3px_4|z'_1}, Z_{x_2|y'_2|z'_2x_3px_4} \in N'$, and $r \in P'$ is a rule (III). Moreover,

$$\begin{array}{ll}
u'Y_{x_1|y'_1x_3px_4|z'_1} \Rightarrow^*_{G'} uA_{x|y_1py_2|z}v_1 & \text{and} \\
Z_{x_2|y'_2|z'_2x_3px_4}v' \Rightarrow^*_{G'} v_2, &
\end{array}$$

where $uA_{x|y_1py_2|z}v_1v_2 = uA_{x|y_1py_2|z}v$. Then, since G' is context-free,

$$\begin{array}{lll}
S_{\varepsilon|\varepsilon|\varepsilon} & \Rightarrow^*_{G'} u'X_{x_1x_2x_3px_4|y'|z'}v' & \\
& \Rightarrow_{G'} u'Y_{x_1|y'_1x_3px_4|z'_1}Z_{x_2|y'_2|z'_2x_3px_4}v' & [r] \\
& \Rightarrow^*_{G'} uA_{x|y_1py_2|z}v_1Z_{x_2|y'_2|z'_2x_3px_4}v' & \\
& \Rightarrow^*_{G'} uA_{x|y_1py_2|z}v & \\
& \Rightarrow^*_{G'} w &
\end{array}$$

which completes the proof, so the claim holds. □

Claim 15. Consider a derivation satisfying claims 12 through 14,

$$\begin{array}{lll}
S_{\varepsilon|\varepsilon|\varepsilon} \Rightarrow^*_{G'} & uA_{\varepsilon|\varepsilon|\varepsilon}v & \\
\Rightarrow_{G'} & uA_{p_1p_2\cdots p_n|\varepsilon|\varepsilon}v & [r] \\
\Rightarrow^*_{G'} & uB_{1\varepsilon|\varepsilon|\varepsilon}B_{2\varepsilon|\varepsilon|\varepsilon}\cdots B_{m\varepsilon|\varepsilon|\varepsilon}v & [\varrho] \\
\Rightarrow^*_{G'} & w &
\end{array}$$

where $w \in T^*$, $u, v \in V'^*$, $A, B_i \in N$, $p_j \in P_{cs}$, $0 \leq i \leq m$, $0 \leq j \leq n$, for some $m \geq 0$, $n \geq 1$, $r \in P'$ is a rule (I), and $\varrho \in P'^*$ contains no rule (I). Then, for every p_j in $A_{p_1 p_2 \cdots p_n|\varepsilon|\varepsilon}$, $1 \leq j \leq n$,

$$\begin{array}{rll} S_{\varepsilon|\varepsilon|\varepsilon} \Rightarrow^*_{G'} & uA_{\varepsilon|\varepsilon|\varepsilon}v & \\ \Rightarrow_{G'} & uA_{p_1 p_2 \cdots p_j \cdots p_n|\varepsilon|\varepsilon}v & [r] \\ \Rightarrow^*_{G'} & uw_1v & [\varrho_1] \\ \Rightarrow_{G'} & uw_2v & [r_1] \\ \Rightarrow^*_{G'} & uw_3v & [\varrho_2] \\ \Rightarrow_{G'} & uw_4v & [r_2] \\ \Rightarrow^*_{G'} & uB_{1\varepsilon|\varepsilon|\varepsilon}B_{2\varepsilon|\varepsilon|\varepsilon} \cdots B_{m\varepsilon|\varepsilon|\varepsilon}v & [\varrho_3] \\ \Rightarrow^*_{G'} & w & \end{array}$$

where $w_1, w_2, w_3, w_4 \in V'^*$, $\varrho = \varrho_1 r_1 \varrho_2 r_2 \varrho_3$, and $r_1 r_2 = \mathrm{perm}(p'_1 p'_2)$, for some rules $p'_1, p'_2 \in P'$ introduced in (IV) of the construction of the form

$$\begin{array}{l} p'_1 \colon X_{x_1|p_j y_1|z_1} \to X_{x_1|y_1|z_1} \\ p'_2 \colon Y_{x_2|y_2|p_j z_2} \to Y_{x_2|y_2|z_2} \end{array}$$

$X_{x_1|p_j y_1|z_1}, X_{x_1|y_1|z_1}, Y_{x_2|y_2|p_j z_2}, Y_{x_2|y_2|z_2} \in N'$.

Proof. By claims 13 and 14,

$$\begin{array}{rll} S_{\varepsilon|\varepsilon|\varepsilon} \Rightarrow^*_{G'} & uA_{\varepsilon|\varepsilon|\varepsilon}v & \\ \Rightarrow_{G'} & uA_{p_1 p_2 \cdots p_j \cdots p_n|\varepsilon|\varepsilon}v & [r] \\ \Rightarrow^*_{G'} & uX_{x_1|y_1 p_j y'_1|z_1}w'Y_{x_2|y_2|z_2 p_j z'_2}v & [\varrho_1] \\ \Rightarrow^*_{G'} & uB_{1\varepsilon|\varepsilon|\varepsilon}B_{2\varepsilon|\varepsilon|\varepsilon} \cdots B_{m\varepsilon|\varepsilon|\varepsilon}v & [\varrho_2] \\ \Rightarrow^*_{G'} & w & \end{array}$$

for some $X_{x_1|y_1 p_j y'_1|z_1}, Y_{x_2|y_2|z_2 p_j z'_2} \in N'$, $w' \in V'^*$, where $\varrho = \varrho_1 \varrho_2$. Observe the rules of G'. Only the rules (IV) are of the form $A_{x|y|z} \to B_{x'|y'|z'}$, where $y > y'$ or $z > z'$; namely, they are of the form $A_{x|p_1 y|p_2 z} \to B_{x|y|z}$, $p_1, p_2 \in P_{cs} \cup \{\varepsilon\}$, $|p_1 p_2| = 1$. In ϱ_2, G' must once get rid of p_j in $X_{x_1|y_1 p_j y'_1|z_1}$ and $Y_{x_2|y_2|z_2 p_j z'_2}$, so $\varrho_2 = \varrho_a r_1 \varrho_b r_2 \varrho_c$, where $r_1 r_2 = \mathrm{perm}(p'_1 p'_2)$ and $p'_1, p'_2 \in P'$ are of the form

$$\begin{array}{l} p'_1 \colon X'_{x_1|p_j y_1|z_1} \to X'_{x_1|y_1|z_1} \\ p'_2 \colon Y'_{x_2|y_2|p_j z_2} \to Y'_{x_2|y_2|z_2} \end{array}$$

for some $X'_{x_1|p_j y_1|z_1}, X'_{x_1|y_1|z_1}, Y'_{x_2|y_2|p_j z_2}, Y'_{x_2|y_2|z_2} \in N'$, and the claim holds. □

The following two claims prove that simulation of two different non-context-free rules of G by G' proceeds properly; informally, two context-free rules simulating one non-context-free never cross with the ones simulating the other one.

Claim 16. Consider a derivation satisfying claims 12 through 15,

$$\begin{array}{rll} S_{\varepsilon|\varepsilon|\varepsilon} \Rightarrow^*_{G'} & uA_{\varepsilon|\varepsilon|\varepsilon}v & \\ \Rightarrow_{G'} & uA_{p_1 p_2 \cdots p_n|\varepsilon|\varepsilon}v & [q] \\ \Rightarrow_{G'} & uX_{x|y_1 r y_2 s y_3|z}v & [\varrho] \\ \Rightarrow^*_{G'} & uB_{1\varepsilon|\varepsilon|\varepsilon}B_{2\varepsilon|\varepsilon|\varepsilon} \cdots B_{m\varepsilon|\varepsilon|\varepsilon}v & [\varphi] \\ \Rightarrow^*_{G'} & w & \end{array}$$

where $w \in T^*$, $u, v \in V'^*$, $A, X, B_i \in N$, $r, s, p_j \in P_{cs}$, $x, y_1, y_2, y_3, z \in (P_{cs} \cup \{\varepsilon\})^k$, $0 \leq i \leq m$, $0 \leq j \leq n$, for some $m \geq 0$, $n \geq 2$, $q \in P'$ is a rule (I), and $\varrho\varphi \in P'^*$ contains no rule (I). Then, there exists a derivation

$$\begin{aligned} S_{\varepsilon|\varepsilon|\varepsilon} &\Rightarrow^*_{G'} uA_{\varepsilon|\varepsilon|\varepsilon}v \\ &\Rightarrow_{G'} uA_{p_1p_2\cdots p_n|\varepsilon|\varepsilon}v && [q] \\ &\Rightarrow_{G'} uX_{x|y_1ry_2sy_3|z}v && [\varrho] \\ &\Rightarrow^*_{G'} uB_{1\varepsilon|\varepsilon|\varepsilon}B_{2\varepsilon|\varepsilon|\varepsilon}\cdots B_{m\varepsilon|\varepsilon|\varepsilon}v && [\varphi'] \\ &\Rightarrow^*_{G'} w \end{aligned}$$

where $\varphi' = \varphi_1 r' \varphi_2 s' \varphi_3$ and

$$\begin{aligned} r'\colon X_{rx_r|ry_r|z_r} &\to X_{rx_r|y_r|z_r} \\ s'\colon X_{sx_s|sy_s|z_s} &\to X_{sx_s|y_s|z_s} \end{aligned}$$

for some $X_{rx_r|ry_r|z_r}$, $X_{rx_r|y_r|z_r}$, $X_{sx_s|sy_s|z_s}$, $X_{sx_s|y_s|z_s} \in N'$.

Proof. Let there exist a derivation

$$\begin{aligned} S_{\varepsilon|\varepsilon|\varepsilon} &\Rightarrow^*_{G'} uA_{\varepsilon|\varepsilon|\varepsilon}v \\ &\Rightarrow_{G'} uA_{p_1p_2\cdots p_n|\varepsilon|\varepsilon}v && [q] \\ &\Rightarrow_{G'} uX_{x|y_1ry_2sy_3|z}v && [\varrho] \\ &\Rightarrow^*_{G'} uB_{1\varepsilon|\varepsilon|\varepsilon}B_{2\varepsilon|\varepsilon|\varepsilon}\cdots B_{m\varepsilon|\varepsilon|\varepsilon}v && [\varphi] \\ &\Rightarrow^*_{G'} w \end{aligned}$$

where $w \in T^*$, $u, v \in V'^*$, $A, X, B_i \in N$, $r, s, p_j \in P_{cs}$, $x, y_1, y_2, y_3, z \in (P_{cs} \cup \{\varepsilon\})^k$, $0 \leq i \leq m$, $0 \leq j \leq n$, for some $m \geq 0$, $n \geq 1$, $q \in P'$ is a rule (I), and $\varrho\varphi \in P'^*$ contains no rule (I). By Claim 15, $\varphi = \varphi_1 p_1 \varphi_2 p_2 \varphi_3$ and $p_1p_2 = \text{perm}(r's')$, where

$$\begin{aligned} r'\colon X_{rx_r|ry_r|z_r} &\to X_{rx_r|y_r|z_r} \\ s'\colon X_{sx_s|sy_s|z_s} &\to X_{sx_s|y_s|z_s} \end{aligned}$$

for some $X_{rx_r|ry_r|z_r}$, $X_{rx_r|y_r|z_r}$, $X_{sx_s|sy_s|z_s}$, $X_{sx_s|y_s|z_s} \in N'$. r' and s' are the rules (IV). The rules (IV) of the form $A_{x|y|z} \to B_{x'|y'|z'}$, where $|y| = |y'| + 1$, process y from left to right direction, so they preserve the order of $c_1c_2\cdots c_n = y$. Also the rules (I), (II), and (V) cannot reorder symbols of y.

Finally, examine the rules (III) of the form

$$A_{x|y|z} \to B_{x_1|y'x_3|z'}B_{x_2|y''|z''x_3},$$

where $y = \text{shuffle}(y'y'')$. Suppose $y = y_1ry_2sy_3$. Then, if both r and s are put into y' or y'', since the shuffle operation preserves the order of symbols, the claim is satisfied. Otherwise, if r is put into y' and s into y'' or r is put into y'' and s into y' and $\varphi = \varphi_1 s \varphi_2 r \varphi_3$, since G' is context-free, there is a derivation $\varphi' = \varphi'_1 r \varphi'_2 s \varphi'_3$ which satisfies the claim. □

Claim 17. Consider a derivation satisfying claims 12 through 15,

$$\begin{aligned} S_{\varepsilon|\varepsilon|\varepsilon} &\Rightarrow^*_{G'} uA_{\varepsilon|\varepsilon|\varepsilon}v \\ &\Rightarrow_{G'} uA_{p_1p_2\cdots p_n|\varepsilon|\varepsilon}v && [q] \\ &\Rightarrow_{G'} uX_{x|y|z_1rz_2sz_3}v && [\varrho] \\ &\Rightarrow^*_{G'} uB_{1\varepsilon|\varepsilon|\varepsilon}B_{2\varepsilon|\varepsilon|\varepsilon}\cdots B_{m\varepsilon|\varepsilon|\varepsilon}v && [\varphi] \\ &\Rightarrow^*_{G'} w \end{aligned}$$

where $w \in T^*$, $u, v \in V'^*$, $A, X, B_i \in N$, $r, s, p_j \in P_{cs}$, $x, z_1, z_2, z_3, z \in (P_{cs} \cup \{\varepsilon\})^k$, $0 \le i \le m$, $0 \le j \le n$, for some $m \ge 0$, $n \ge 1$, $q \in P'$ is a rule (I), and $\varrho\varphi \in P'^*$ contains no rule (I). Then, there exists a derivation

$$\begin{array}{rll} S_{\varepsilon|\varepsilon|\varepsilon} \Rightarrow^*_{G'} & uA_{\varepsilon|\varepsilon|\varepsilon}v & \\ \Rightarrow_{G'} & uA_{p_1p_2\cdots p_n|\varepsilon|\varepsilon}v & [q] \\ \Rightarrow_{G'} & uX_{x|y|z_1rz_2sz_3}v & [\varrho] \\ \Rightarrow^*_{G'} & uB_{1\varepsilon|\varepsilon|\varepsilon}B_{2\varepsilon|\varepsilon|\varepsilon}\cdots B_{m\varepsilon|\varepsilon|\varepsilon}v & [\varphi'] \\ \Rightarrow^*_{G'} & w & \end{array}$$

where $\varphi' = \varphi_1 r' \varphi_2 s' \varphi_3$ and

$$\begin{array}{l} r'\colon X_{rx_r|y_r|rz_r} \rightarrow X_{rx_r|y_r|z_r} \\ s'\colon X_{sx_s|y_s|sz_s} \rightarrow X_{sx_s|y_s|z_s} \end{array}$$

for some $X_{rx_r|y_r|z_r}$, $X_{rx_r|y_r|rz_r}$, $X_{sx_s|y_s|z_s}$, $X_{sx_s|y_s|sz_s} \in N'$.

Proof. Prove by analogy with Claim 16. □

Claim 18. Every $x \in \mathscr{F}(G')$, where $x \Rightarrow^*_{G'} w$ and $w \in T^*$, can be derived in G' as follows.

$$S_{\varepsilon|\varepsilon|\varepsilon} = x_0 \Rightarrow^{d_1}_{G'} x_1 \Rightarrow^{d_2}_{G'} x_2 \Rightarrow^{d_3}_{G'} \cdots \Rightarrow^{d_{h-1}}_{G'} x_{h-1} \Rightarrow^{d_h}_{G'} x_h = x,$$

for some $h \ge 0$, where $d_i \in \{1, 2\}$, $1 \le i \le h$, so that

(1) if $d_i = 1$, then $x_{i-1} = uA_{x|y|z}v$, $x_i = uzv$, $x_{i-1} \Rightarrow_{G'} x_i$ $[A_{x|y|z} \rightarrow z]$, where $u, v \in V'^*$, $z \in \{B_{x|y|z}, C_{x_C|y_C|z_C}D_{x_D|y_D|z_D}, a\}$, for some $a \in (T \cup \{\varepsilon\})$, $A_{x|y|z}$, $B_{x|y|z}$, $C_{x_C|y_C|z_C}$, $D_{x_D|y_D|z_D} \in N'$;

(2) if $d_i = 2$, then $x_{i-1} = uA_{x|py|z}vB_{x'|y'|pz'}w$, $x_i = uC_{x|y|z}vD_{x'|y'|z'}w$, and

$$\begin{array}{rll} uA_{x|py|z}vB_{x'|y'|pz'}w & \Rightarrow_{G'} uC_{x|y|z}vB_{x'|y'|pz'}w & [A_{x|py|z} \rightarrow C_{x|y|z}] \\ & \Rightarrow_{G'} uC_{x|y|z}vD_{x'|y'|z'}w & [B_{x'|y'|pz'} \rightarrow D_{x'|y'|z'}] \end{array}$$

for some $u, v, w \in V'^*$, $A_{x|py|z}, B_{x'|y'|pz'}, C_{x|y|z}, D_{x'|y'|z'} \in N'$, and

$$p\colon (A, B) \rightarrow (C, D) \in P.$$

Proof. Follows from claims 13 through 17. □

Define the finite substitution γ from V^* to V'^*as

$$\gamma(A) = \{A_{x|y|z} \mid x, y, z \in (P \cup \{\varepsilon\})^k\},$$

for $A \in N$, and $\gamma(a) = a$ otherwise. Let γ^{-1} be the inverse of γ. Next, we proof $\mathrm{L}(G) = \mathrm{L}(G')$ by establishing Claims 19 and 20.

Claim 19. If $S \Rightarrow^m_G w \Rightarrow^*_G \bar{w}$, where $m \ge 0$, $w \in V^*$, and $\bar{w} \in T^*$, then $S_{\varepsilon|\varepsilon|\varepsilon} \Rightarrow^*_{G'} w'$, where $w' \in V'^*$ and $w' \in \gamma(w)$.

Proof. We prove the claim by induction on $m \geq 0$.

Basis. Let $m = 0$. That is, $S \Rightarrow_G^0 S \Rightarrow_G^* \bar{w}$, for some $\bar{w} \in T^*$. Clearly, $S_{\varepsilon|\varepsilon|\varepsilon} \Rightarrow_{G'}^0 S_{\varepsilon|\varepsilon|\varepsilon}$, where $S_{\varepsilon|\varepsilon|\varepsilon} \in \gamma(S)$, so the basis holds.

Induction Hypothesis. Suppose that there exists $n \geq 0$ such that Claim 19 holds for all m with $0 \leq m \leq n$.

Induction Step. Let $S \Rightarrow_G^{n+1} w \Rightarrow_G^* \bar{w}$, for some $w \in V^*$ and $\bar{w} \in T^*$. Then, $S \Rightarrow_G^n v \Rightarrow_G w \Rightarrow_G^* \bar{w}$, where $v \in V^*$, and there exists $p \in P$ such that $v \Rightarrow_G w$ $[p]$. By the induction hypothesis, $S_{\varepsilon|\varepsilon|\varepsilon} \Rightarrow_{G'}^* v'$, where $v' \in \gamma(v)$. Next, we consider the following three forms of p according to the binary form of G.

(i) Let p be of the form $p\colon (A) \rightarrow (X)$. Then, $v = u_1 A u_2$, $u_1, u_2 \in V^*$, and $u_1 A u_2 \Rightarrow_G u_1 X u_2$ $[p]$, where $u_1 X u_2 = w$. Since $S_{\varepsilon|\varepsilon|\varepsilon} \Rightarrow_{G'}^* v'$, where $v' \in \gamma(u_1)\{A_{x|y|z}\}\gamma(u_2)$, $v' \in \gamma(v)$, for some $x, y, z \in (P_{cs} \cup \{\varepsilon\})^k$. Without any loss of generality, suppose that x, y, z are correct; we can obviously make this assumption since the rules of G' cover all possibilities. Consider the following two cases depending on whether $X \in N$.

  (a) Let $X \in N$.

    (1) Suppose A corresponds to the root node of a subgraph t_i in some h-division t' of tree $t = {}_G\triangle(S \Rightarrow_G^* \bar{w})$, where $1 \leq h \leq k$, $1 \leq i \leq h$, which satisfies Theorem 6.3.1. Then, $x = y = z = \varepsilon$ and, by the construction of G', there exists a rule $A_{\varepsilon|\varepsilon|\varepsilon} \Rightarrow_{G'} A_{\bar{x}|\varepsilon|\varepsilon} \in P'$ introduced in (I), where $\bar{x} = p_1 p_2 \cdots p_l \in P^*$, $0 \leq l \leq k$. Without any loss of generality, suppose that $p_1 p_2 \cdots p_l$ corresponds to all non-context-free rules of t_i and they are in a proper order; since rules introduced in (I) cover all possibilities, we can make this assumption. By this rule $v' \Rightarrow_{G'} v''$ and since $\gamma^{-1}(A_{\varepsilon|\varepsilon|\varepsilon}) = \gamma^{-1}(A_{\bar{x}|\varepsilon|\varepsilon})$, $v'' \in \gamma(v)$.

    (2) Let $\bar{v} = v''$ if (1) is performed and $\bar{v} = v'$ otherwise. Consequently, $\bar{v} \in \gamma(u_1)\{A_{x'|y'|z'}\}\gamma(u_2)$, for some $A_{x'|y'|z'} \in N'$, and, by the construction of G', there exists a nonterminal $X_{x'|y'|z'} \in N'$ and a rule $A_{x'|y'|z'} \rightarrow X_{x'|y'|z'} \in P'$ by which $\bar{v} \Rightarrow_{G'} w'$, where $w' \in \gamma(u_1)\{X_{x'|y'|z'}\}\gamma(u_2)$, and since $X_{x'|y'|z'} \in \gamma(X)$, $w' \in \gamma(u_1 X u_2)$ and the claim holds.

  (b) Let $X \notin N$; that is $X \in T$. Then, $x = y = z = \varepsilon$ and, by the construction of G', there exists a rule $A_{\varepsilon|\varepsilon|\varepsilon} \rightarrow X \in P'$ by which $v' \Rightarrow_{G'} w'$, where $w' \in \gamma(u_1)\{X\}\gamma(u_2)$. Since $X \in T$, $\gamma(X) = X$ and $\gamma(u_1)\{X\}\gamma(u_2) = \gamma(u_1 X u_2)$ which satisfies the claim.

  Note that even if A corresponds to the root node of a subgraph t_i in some h-division t' of tree $t = {}_G\triangle(S \Rightarrow_G^* \bar{w})$, where $1 \leq h \leq k$, $1 \leq i \leq h$, according to Theorem 6.3.1, t'_i is obviously without any context-dependent nodes since it contains only one terminal node.

(ii) Let p be of the form $p\colon (A) \rightarrow (BC)$. Then, $v = u_1 A u_2$, $u_1, u_2 \in V^*$, and $u_1 A u_2 \Rightarrow_G u_1 BC u_2$ $[p]$, where $u_1 BC u_2 = w$. Since $S_{\varepsilon|\varepsilon|\varepsilon} \Rightarrow_{G'}^* v'$, where $v' \in \gamma(v)$, $v' \in \gamma(u_1)\{A_{x|y|z}\}\gamma(u_2)$, for some $x, y, z \in (P_{cs} \cup \{\varepsilon\})^k$. Without any loss of generality, suppose that x, y, z are correct; we can obviously make this assumption since the rules of G' cover all possibilities.

(1) Suppose A corresponds to the root node of a subgraph t_i in some h-division t' of tree $t = {}_G\triangle(S \Rightarrow_G^* \bar{w})$, where $1 \leq h \leq k$, $1 \leq i \leq h$, which satisfies Theorem 6.3.1. Then, $x = y = z = \varepsilon$ and, by the construction of G', there exists a rule $A_{\varepsilon|\varepsilon|\varepsilon} \Rightarrow_{G'} A_{\bar{x}|\varepsilon|\varepsilon} \in P'$ introduced in (I), where $\bar{x} = p_1p_2\cdots p_l \in P^*$, $0 \leq l \leq k$. Without any loss of generality, suppose that $p_1p_2\cdots p_l$ corresponds to all non-context-free rules of t_i and they are in a proper order; since rules introduced in (I) cover all possibilities, we can make this assumption. By this rule $v' \Rightarrow_{G'} v''$ and since $\gamma^{-1}(A_{\varepsilon|\varepsilon|\varepsilon}) = \gamma^{-1}(A_{\bar{x}|\varepsilon|\varepsilon})$, $v'' \in \gamma(v)$.

(2) Let $\bar{v} = v''$ if (1) is performed and $\bar{v} = v'$ otherwise. Then, $\bar{v} \in \gamma(u_1)\{A_{x'|y'|z'}\}\gamma(u_2)$, for some $A_{x'|y'|z'} \in N'$, and, by the construction of G', there exist two nonterminals $B_{x''|y''|z''}, C_{x'''|y'''|z'''} \in N'$ and a rule $A_{x'|y'|z'} \to B_{x''|y''|z''}C_{x'''|y'''|z'''} \in P'$ by which $\bar{v} \Rightarrow_{G'} w'$, where $w' \in \gamma(u_1)\{B_{x''|y''|z''}\}\{C_{x'''|y'''|z'''}\}\gamma(u_2)$, and since $B_{x''|y''|z''} \in \gamma(B)$ and $C_{x'''|y'''|z'''} \in \gamma(C)$, $w' \in \gamma(u_1BCu_2)$ and the claim holds.

(iii) Let p be of the form $p\colon (A, B) \to (C, D)$. Then, $v = u_1Au_2Bu_3$, $u_1, u_2, u_3 \in V^*$, and $u_1Au_2Bu_3 \Rightarrow_G u_1Cu_2Du_3$ [p], where $u_1Cu_2Du_3 = w$. Since $S_{\varepsilon|\varepsilon|\varepsilon} \Rightarrow_{G'}^* v'$, where $v' \in \gamma(v)$, $v' \in \gamma(u_1)\{A_{x_A|y_A|z_A}\}\gamma(u_2)\{B_{x_B|y_B|z_B}\}\gamma(u_3)$, for some $x_A, y_A, z_A, x_B, y_B, z_B \in (P_{cs} \cup \{\varepsilon\})^k$. Without any loss of generality, suppose that $x_A, y_A, z_A, x_B, y_B, z_B$ are correct; we can obviously make this assumption since the rules of G' cover all possibilities. By the construction of G', there exist two nonterminals $C_{x_C|y_C|z_C}$, $D_{x_D|y_D|z_D} \in N'$ and two rules

$$\begin{aligned} &A_{x_A|y_A|z_A} \to C_{x_C|y_C|z_C}, \\ &B_{x_B|y_B|z_B} \to D_{x_D|y_D|z_D} \quad \in P' \end{aligned}$$

by which $v' \Rightarrow_{G'}^2 w'$, where $w' \in \gamma(u_1)\{C_{x_C|y_C|z_C}\}\gamma(u_2)\{D_{x_D|y_D|z_D}\}\gamma(u_3)$, and since $C_{x_C|y_C|z_C} \in \gamma(C)$ and $D_{x_D|y_D|z_D} \in \gamma(D)$, $w' \in \gamma(u_1Cu_2Du_3)$ and the claim holds.

Note that A and B cannot correspond to the root node of any subgraph t_i in any h-division t' of tree $t = {}_G\triangle(S \Rightarrow_G^* \bar{w})$, where $1 \leq h \leq k$, $1 \leq i \leq h$, according to Theorem 6.3.1 since A and B are context-dependent.

We covered all possible forms of p and Claim 19 holds. □

Claim 20. Let $S_{\varepsilon|\varepsilon|\varepsilon} \Rightarrow_{G'}^{d_1} x_1 \Rightarrow_{G'}^{d_2} \cdots \Rightarrow_{G'}^{d_{m-1}} x_{m-1} \Rightarrow_{G'}^{d_m} x_m$ *be a derivation that satisfies Claim 18, for some* $m \geq 0$. *Then* $S \Rightarrow_G^* w$, *where* $x_m \in \gamma(w)$.

Without any loss of generality, suppose G' satisfies Claim 8 through 18.

Proof. We prove the claim by induction on $m \geq 0$.

Basis. Let $m = 0$. That is, $S_{\varepsilon|\varepsilon|\varepsilon} \Rightarrow_{G'}^0 S_{\varepsilon|\varepsilon|\varepsilon}$ and, clearly, $S \Rightarrow_G^0 S$, where $S_{\varepsilon|\varepsilon|\varepsilon} \in \gamma(S)$, so the basis holds.

Induction Hypothesis. Suppose that there exists $n \geq 0$ such that Claim 19 holds for all m with $0 \leq m \leq n$.

Induction Step. Let

$$S_{\varepsilon|\varepsilon|\varepsilon} \Rightarrow_{G'}^{d_1} x_1 \Rightarrow_{G'}^{d_2} \cdots \Rightarrow_{G'}^{d_{n-1}} x_{n-1} \Rightarrow_{G'}^{d_n} x_n \Rightarrow_{G'}^{d_{n+1}} x_{n+1},$$

for some $x_i \in V^*$ and $d_i \in \{1, 2\}$, $1 \leq i \leq n+1$. By the induction hypothesis, $S \Rightarrow_G^* v$, where $x_n \in \gamma(v)$. Let us divide the proof into two cases depending on $d_{n+1} \in \{1, 2\}$.

(i) Let $d_{n+1} = 1$. Then, $x_n \Rightarrow_{G'}^{d_{n+1}} x_{n+1}$ by some rule $p'\colon A_{x|y|z} \rightarrow X \in P'$, $x_n = u_1 A_{x|y|z} u_2$, $x_{n+1} = u_1 X u_2$, for some $A_{x|y|z} \in N'$ and $X, u_1, u_2 \in V^*$, and $v = \gamma^{-1}(u_1 A_{x|y|z} u_2) = \gamma^{-1}(u_1) A \gamma^{-1}(u_2)$. Next, we consider the following four forms of p' according to the construction of G'.

   (a) Let p' be of the form $p'\colon A_{\varepsilon|\varepsilon|\varepsilon} \rightarrow A_{\bar{x}|\varepsilon|\varepsilon}$, for some $\bar{x} \in (P_{cs} \cup \{\varepsilon\})^k$; obviously, $x = y = z = \varepsilon$ and $X = A_{\bar{x}|\varepsilon|\varepsilon}$. However, since $\gamma^{-1}(A_{\varepsilon|\varepsilon|\varepsilon}) = \gamma^{-1}(A_{\bar{x}|\varepsilon|\varepsilon})$ and $x_n \in \gamma(v)$, $x_{n+1} \in \gamma(v)$ and the claim holds trivially.

   (b) Let p' be of the form $p'\colon A_{x|y|z} \rightarrow B_{x|y|z}$, for some $B_{x|y|z} \in N'$; $X = B_{x|y|z}$. By the construction of G', p' was introduced according to some $p\colon (A) \rightarrow (B) \in P$ by which
   $$\gamma^{-1}(u_1) A \gamma^{-1}(u_2) \Rightarrow_G \gamma^{-1}(u_1) B \gamma^{-1}(u_2).$$
   Since $B_{x|y|z} \in \gamma(B)$ and
   $$\gamma(\gamma^{-1}(u_1) B \gamma^{-1}(u_2)) = \{u_1\}\gamma(B)\{u_2\},$$
   $u_1 B_{x|y|z} u_2 \in \gamma(\gamma^{-1}(u_1) B \gamma^{-1}(u_2))$ and the claim holds.

   (c) Let p' be of the form $p'\colon A_{x|y|z} \rightarrow B_{x'|y'|z'} C_{x''|y''|z''}$, for some $B_{x'|y'|z'}, C_{x''|y''|z''} \in N'$; $X = B_{x'|y'|z'} C_{x''|y''|z''}$. By the construction of G', p' was introduced according to some $p\colon (A) \rightarrow (BC) \in P$ by which
   $$\gamma^{-1}(u_1) A \gamma^{-1}(u_2) \Rightarrow_G \gamma^{-1}(u_1) BC \gamma^{-1}(u_2).$$
   Since $B_{x'|y'|z'} C_{x''|y''|z''} \in \gamma(BC)$ and
   $$\gamma(\gamma^{-1}(u_1) BC \gamma^{-1}(u_2)) = \{u_1\}\gamma(BC)\{u_2\},$$
   $u_1 B_{x'|y'|z'} C_{x''|y''|z''} u_2 \in \gamma(\gamma^{-1}(u_1) BC \gamma^{-1}(u_2))$ and the claim holds.
   Note that the correctness of $x, y, z, x', y', z', x'', y'', z''$ in
   $$p'\colon A_{x|y|z} \rightarrow B_{x'|y'|z'} C_{x''|y''|z''}$$
   follows from Claim 8 through 18.

   (d) Let p' be of the form $p'\colon A_{\varepsilon|\varepsilon|\varepsilon} \rightarrow a$, for some $a \in T$; obviously, $x = y = z = \varepsilon$ and $X = a$. By the construction of G', p' was introduced according to some $p\colon (A) \rightarrow (a) \in P$ by which $\gamma^{-1}(u_1) A \gamma^{-1}(u_2) \Rightarrow_G \gamma^{-1}(u_1) a \gamma^{-1}(u_2)$. Since $a = \gamma(a)$ and $\gamma(\gamma^{-1}(u_1) a \gamma^{-1}(u_2)) = \{u_1 a u_2\}$, $u_1 a u_2 \in \gamma(\gamma^{-1}(u_1) a \gamma^{-1}(u_2))$ and the claim holds.

(ii) Let $d_{n+1} = 2$. Then, $x_n \Rightarrow_{G'}^{d_{n+1}} x_{n+1}$ by two rules
$$\begin{aligned} p_1'\colon & A_{x_A|y_A|z_A} \rightarrow C_{x_C|y_C|z_C}, \\ p_2'\colon & B_{x_B|y_B|z_B} \rightarrow D_{x_D|y_D|z_D} \quad \in P', \end{aligned}$$
$x_n = u_1 A_{x_A|y_A|z_A} u_2 B_{x_B|y_B|z_B} u_3$, and $x_{n+1} = u_1 C_{x_C|y_C|z_C} u_2 D_{x_D|y_D|z_D} u_3$, for some $A_{x_A|y_A|z_A}, B_{x_B|y_B|z_B}, C_{x_C|y_C|z_C}, D_{x_D|y_D|z_D} \in N'$ and $u_1, u_2, u_3 \in V^*$. Then, $\gamma^{-1}(x_n) = \gamma^{-1}(u_1) A \gamma^{-1}(u_2) B \gamma^{-1}(u_3) = v$ and, by the construction of G', p_1' and p_2' were introduced based on some rule $p\colon (A, B) \rightarrow (C, D) \in P$ by which
$$\gamma^{-1}(u_1) A \gamma^{-1}(u_2) B \gamma^{-1}(u_3) \Rightarrow_G \gamma^{-1}(u_1) C \gamma^{-1}(u_2) D \gamma^{-1}(u_3).$$

Since $C_{x_C|y_C|z_C} \in \gamma(C)$, $D_{x_D|y_D|z_D} \in \gamma(D)$, and

$$\gamma(\gamma^{-1}(u_1)C\gamma^{-1}(u_2)D\gamma^{-1}(u_3)) = \{u_1\}\gamma(C)\{u_2\}\gamma(D)\{u_3\},$$

$u_1 C_{x_C|y_C|z_C} u_2 D_{x_D|y_D|z_D} u_3 \in \gamma(\gamma^{-1}(u_1)C\gamma^{-1}(u_2)D\gamma^{-1}(u_3))$ and the claim holds.

We covered all possible forms of d_{n+1} and Claim 20 holds. $\square$

Finally, we establish $\mathrm{L}(G) = \mathrm{L}(G')$. Consider Claim 19 with $w \in T^*$. Then, $S \Rightarrow_G^* w$ implies that $S_{\varepsilon|\varepsilon|\varepsilon} \Rightarrow_{G'}^* w'$, where $\gamma(w') = w$, and since for $w \in T^*$ it holds that $\gamma(w) = w$, $w' = w$. Thus, $\mathrm{L}(G) \subseteq \mathrm{L}(G')$. Consider Claim 20 with $x_m \in T^*$. Then, $S_{\varepsilon|\varepsilon|\varepsilon} \Rightarrow_{G'}^* x_m$ implies that $S \Rightarrow_G^* w$, where $\gamma(x_m) = w$, and since for $x_m \in T^*$, it holds that $\gamma(x_m) = x_m = w$. Thus, $\mathrm{L}(G') \subseteq \mathrm{L}(G)$. Hence, $\mathrm{L}(G) = \mathrm{L}(G')$, which completes the if part of the proof and Theorem 6.3.1 holds. $\square$

Later in Chapter 7 we demonstrate how to utilize the established result in practice to obtain a positive proof that some language is context-free.

6.4 Cooperating Distributed Grammar Systems

Alongside with parallel processing of information discussed in the previous section in sense of scattered context grammars, lot of computational systems nowadays consist of numerous processors distributed over a long distance. Cooperating distributed grammar systems (see Section 2.2) introduced in [11] which this section focuses on may stand as an appropriate model to represent this kind of computation.

In essence, a CDGS is an n-tuple of context-free grammars which alternate generating the common sentence under the conditions given by derivation mode; one component starts generation from the start symbol, then, the other continues with the sentence form generated by the first one, etc. Depending on the used derivation mode their power ranges from context-free grammars up to matrix grammars or extended tabled zero-sided Lindenmayer systems. Let us investigate the derivation process of a CDGS from the derivation-tree point of view. The derivation tree is composed of several layers in the top-down way, where each of them corresponds to the part of the derivation performed by one of the components. A change of the component corresponds to a certain cut of the derivation tree. Next, we prove that if we limit the possible number of these cuts in every derivation tree of a CDGS it in fact generates a context-free languages. Most importantly, this gives us necessary but also sufficient conditions for a language to be context-free.

For a brevity, for a CDGS G, component i, and a derivation mode t, we write $\Rightarrow_i^t$ instead of $\Rightarrow_{G(i)}^t$, in what follows, if there is no risk of confusion.

Theorem 6.4.1. *A language L is context-free iff there is a constant $k \geq 0$ and a cooperating distributed grammar system $G = (V, T, S, P_1, \dots, P_n)$ of degree $n \geq 1$ such that $L = \mathrm{L}(G^t)$, and for every $w \in \mathrm{L}(G^t)$, there exists a derivation d*

$$d = S \Rightarrow_{i_1}^t \cdots \Rightarrow_{i_m}^t w$$

in G of a length $m \geq 1$, where $1 \leq i_j \leq n$, $1 \leq j \leq m$, with ${}_G\triangle(d)$ in which there exists k or fewer component-change cuts.

Proof. Construction. Let $k \geq 0$ be a constant. Let $G = (V, T, S, P_1, \dots, P_n)$ be a CDGS of degree $n \geq 1$ such that for every $w \in \mathrm{L}(G^t)$, there exists a derivation d

$$d = S \Rightarrow_{i_1}^{t} \cdots \Rightarrow_{i_m}^{t} w$$

in G of a length $m \geq 1$, where $1 \leq i_j \leq n$, $1 \leq j \leq m$, with ${}_G\triangle(d)$ in which there exists k or fewer component-change cuts. Without any loss of generality, consider $n = 3$ (see Theorem 2.2.2)—for $n \in \{1, 2\}$, P_2 or P_3 is empty. Define CFG $G' = (V', T, P', S')$ as follows. Initially, set

$$N' = \{\langle A, i, x\rangle \mid A \in N, i \in \{1, 2, 3\}, x \in \{1, 2, 3, \varepsilon\}^k\} \cup \{S'\}$$

and $P' = \emptyset$. Define the function $\mathcal{D}\colon N' \times V \to N' \cup T$, as $\mathcal{D}(\langle A, i, x\rangle, a) = a$, for $a \in T$, $\mathcal{D}(\langle A, i, x\rangle, B) = \langle B, i, x\rangle$, for $B \in N$. Define the morphism $\gamma\colon N' \cup T \to N \cup T$ as $\gamma(a) = a$, for $a \in T$, $\gamma(\langle A, i, x\rangle) = A$, for $\langle A, i, x\rangle \in N'$. Construct P' by performing (I) through (III) given next.

(I) For all $i \in \{1, 2, 3\}$, and $x \in \{1, 2, 3, \varepsilon\}^k$, where $S \in \mathrm{dom}(P_i)$, add

$$S' \to \langle S, i, x\rangle \text{ to } P';$$

(II) for all $i \in \{1, 2, 3\}$, $A \to w$ in P_i, and $x \in \{1, 2, 3, \varepsilon\}^k$, where $w = a_1 a_2 \cdots a_z$, for some $z \geq 0$, $a_j \in N \cup T$, and $0 \leq j \leq z$, add

$$\langle A, i, x\rangle \to \mathcal{D}(\langle A, i, x\rangle, a_1)\mathcal{D}(\langle A, i, x\rangle, a_2) \cdots \mathcal{D}(\langle A, i, x\rangle, a_z) \text{ to } P';$$

(III) for all $i \in \{1, 2, 3\}$, $A \notin \mathrm{dom}(P_i)$, and $x \in \{1, 2, 3, \varepsilon\}^k$, where $x = ax'$, for some $a \in \{1, 2, 3\}$, add

$$\langle A, i, x\rangle \to \langle A, a, x'\rangle \text{ to } P'.$$

Basic Idea. Every nonterminal $\langle A, i, x\rangle \in N'$ encodes the corresponding nonterminal $A \in N$, currently active component i, and a string of the following active components in their precise order, x. G' simulates the derivations in G as follows. By some initial rule (I), a first active component i and an order of the following active components x are nondeterministically chosen. While i is an active component, the rules (II) are applied rewriting all the possible nonterminals according to P_i; since G' is context-free, without any loss of generality we can reorder derivation steps in this way. When there is no applicable rule in P_i, G' simulates a change of a component by the rules (III).

Claim 21. For every derivation

$$S \Rightarrow_{i_1}^{t} \cdots \Rightarrow_{i_m}^{t} w$$

in G of a length $m \geq 0$, for $1 \leq i_j \leq n$, $1 \leq j \leq m$, there exists a derivation $S \Rightarrow^* w'$ in G', where $\gamma(w') = w$.

We prove Claim 21 by induction on m.

Proof. Basis. If $m = 0$, $S = w$. Then, by some rule (I), $S' \Rightarrow \langle S, i, x\rangle$, where $\gamma(\langle S, i, x\rangle) = S$, and the claim holds.

Induction Hypothesis. Suppose there exists ℓ such that the claim holds for all $0 \leq m \leq \ell$.

Induction Step. Consider a derivation $S \Rightarrow_{i_1}^t \cdots \Rightarrow_{i_{\ell+1}}^t w$ in G, for some $1 \leq i_{\ell+1} \leq n$. Since $\ell + 1 \geq 1$, we can express the derivation as

$$S \Rightarrow_{i_1}^t \cdots \Rightarrow_{i_\ell}^t v \Rightarrow_{i_{\ell+1}}^t w.$$

By the induction hypothesis $S \Rightarrow^* v'$ in G', where $\gamma(v') = v$. Then, v is of the form $u_0 A_1 u_1 \cdots A_z u_z$, $u_0, u_j \in T^*$, $A_j \in N$, $1 \leq j \leq z$, and v' is of the form

$$u_0 \langle A_1, i, x \rangle u_1 \cdots \langle A_z, i, x \rangle u_z.$$

Without any loss of generality, suppose x corresponds to the following active components of G. If $i \neq i_{\ell+1}$, $x = i_{\ell+1} x'$ and by the rules (III)

$$u_0 \langle A_1, i, x \rangle u_1 \cdots \langle A_z, i, x \rangle u_z \Rightarrow^z u_0 \langle A_1, i_{\ell+1}, x' \rangle u_1 \cdots \langle A_z, i_{\ell+1}, x' \rangle u_z;$$

otherwise, $x' = x$. Express $v \Rightarrow_{i_{\ell+1}}^t w$ as

$$v = w_0 \Rightarrow w_1 \; [p_1] \Rightarrow w_2 \; [p_2] \Rightarrow \cdots \Rightarrow w_o \; [p_o] = w,$$

where $p_j \in P_{i_{\ell+1}}$, for $1 \leq j \leq o$, $o \geq 0$. Suppose there exists w_j, $1 \leq j \leq o-1$, where $S \Rightarrow^* w'_j$ in G', $\gamma(w'_j) = w_j$, and

$$w_j = w_{j_1} A w_{j_2} \Rightarrow w_{j_1} y w_{j_2} \; [p_{j+1}] = w_{j+1}$$

by some rule $p_{j+1} \colon A \to y \in P_{i_{\ell+1}}$. Then, $w'_j = w'_{j_1} \langle A, i_{\ell+1}, x' \rangle w'_{j_2}$, $y = a_1 a_2 \cdots a_r$, for some $r \geq 0$, and by the construction of G' there exists a rule (II)

$$\langle A, i_{\ell+1}, x' \rangle \to \mathcal{D}(\langle A, i_{\ell+1}, x' \rangle, a_1) \mathcal{D}(\langle A, i_{\ell+1}, x' \rangle, a_2) \cdots \mathcal{D}(\langle A, i_{\ell+1}, x' \rangle, a_r) = y',$$

where $\gamma(y') = y$. Then, by this rule $w'_j \Rightarrow w'_{j+1}$, where $\gamma(w'_{j+1}) = w_{j+1}$. Notice that for $w_0 = w_j$, $S \Rightarrow^* w'_j$ holds trivially by the induction basis which completes the proof. $\square$

Claim 22. Every derivation of $x \in \mathrm{L}(G')$ can be derived in G' as follows.

$$S' = x_0 \Rightarrow^{d_1} x_1 \Rightarrow^{d_2} x_2 \Rightarrow^{d_3} \cdots \Rightarrow^{d_{h-1}} x_{h-1} \Rightarrow^{d_h} x_h = x,$$

for some $h \geq 0$, where $d_i \in \{I, II, III\}$, $1 \leq i \leq h$, so that

1. *if $d_i = I$, then $x_{i-1} = S'$, $x_i = \langle S, j, y \rangle$, for some $1 \leq j \leq n$ and $y \in \{1, 2, 3, \varepsilon\}^k$, and by some rule (I), $S' \Rightarrow \langle S, j, y \rangle$.*

2. *if $d_i = II$, then*

$$x_{i-1} = u_0 \langle A_1, j, y \rangle u_1 \langle A_2, j, y \rangle u_2 \cdots u_{z-1} \langle A_z, j, y \rangle u_z,$$

where $u_0, u_l \in T^*, \langle A_l, j, y \rangle \in N'$, $1 \leq l \leq z$, for some $1 \leq j \leq n$, $y \in \{1, 2, 3, \varepsilon\}^k$, and $z \geq 0$, and $x_{i-1} \Rightarrow^* x_i$ only by the rules (II) of the form $\langle A, j, y \rangle \to X$ so that there is no $\langle B, j, y \rangle \in \mathrm{alph}(x_i)$, where $\langle B, j, y \rangle \to Y$ is a rule (II), for any $A, B \in N$, $X, Y \in V'^*$.

3. *if $d_i = III$, then*

$$x_{i-1} = u_0 \langle A_1, j, y \rangle u_1 \langle A_2, j, y \rangle u_2 \cdots u_{z-1} \langle A_z, j, y \rangle u_z,$$

where $u_0, u_l \in T^*, \langle A_l, j, y \rangle \in N'$, $1 \leq l \leq z$, for some $1 \leq j \leq n$, $y \in \{1, 2, 3, \varepsilon\}^k$, $z \geq 0$, and $y = a y'$, where $a \in \{1, 2, 3\}$, and

$$x_{i-1} \Rightarrow^z x_i = u_0 \langle A_1, a, y' \rangle u_1 \langle A_2, a, y' \rangle u_2 \cdots u_{z-1} \langle A_z, a, y' \rangle u_z$$

is obtained by the rules (III) of the form $\langle A_l, j, y \rangle \to \langle A_l, a, y' \rangle$.

Proof. Let us inspect the rules of G'. Since S' is the start symbol and never occurs on the right-hand side of any rule, 1. is trivial. Observe (II) and (III) of construction; the rules are designed that the left hand sides of (II) and (III) are mutually exclusive. In (II), while rewriting $\langle A, i, x\rangle$, each right-hand side nonterminal inherits i and x and, therefore, until there is an applicable rule (II), it cannot be rewritten by any rule (III). Since G' is a CFG, without any loss of generality, we can reorder any derivation to first apply all possible rules (II) before applying any rule (III) which completes 2. Finally, if there is no applicable rule (II), all nonterminals are of the form $\langle A, i, x\rangle$ with same i and x. If $x = ax'$, for some $a \in \{1, 2, 3\}$, all nonterminals can be rewritten by the rules (III) which completes 3. The rigorous proof is left to the reader. $\square$

Claim 23. For every derivation

$$S' = x_0 \Rightarrow^{d_1} x_1 \Rightarrow^{d_2} x_2 \Rightarrow^{d_3} \cdots \Rightarrow^{d_{h-1}} x_{h-1} \Rightarrow^{d_h} x_h = w',$$

in G', for some $h \geq 1$, where d_i satisfies Claim 22, and $1 \leq i \leq h$, there is a derivation

$$S \Rightarrow^t_{i_1} \cdots \Rightarrow^t_{i_m} w$$

in G of a length $m \geq 0$, for $1 \leq i_j \leq n$, $1 \leq j \leq m$, where $\gamma(w') = w$.

We proof Claim 23 by induction on h.

Proof. Basis. If $h = 1$, $S' \Rightarrow^I \langle S, i, x\rangle = w'$, for some $i \in \{1, 2, 3\}$ and $x \in \{1, 2, 3, \varepsilon\}^k$. Then, with $S = w$, $\gamma(w') = w$, so the basis holds.

Induction Hypothesis. Suppose there exists ℓ such that the claim holds for all $1 \leq h \leq \ell$.

Induction Step. Consider a derivation

$$S' \Rightarrow^{d_1} \cdots \Rightarrow^{d_{\ell+1}} x_{\ell+1} = w',$$

in G'. Since $\ell + 1 \geq 2$, there exists $v' = x_\ell$, where

$$S' \Rightarrow^{d_1} \cdots \Rightarrow^{d_\ell} x_\ell \Rightarrow^{d_{\ell+1}} x_{\ell+1}.$$

By the induction hypothesis, $S \Rightarrow^t_{i_1} \cdots \Rightarrow^t_{i_m} v$, where $\gamma(v') = v$, for some $m \geq 0$. Next, we consider two cases depending on $d_{\ell+1}$.

a) If $d_{\ell+1} = II$, $x_\ell \Rightarrow^{d_{\ell+1}} x_{\ell+1}$ can be expressed as

$$x_\ell = x_0 \Rightarrow x_1\ [p_1] \Rightarrow x_2\ [p_2] \Rightarrow \cdots \Rightarrow x_{z-1}\ [p_{z-1}] \Rightarrow x_z\ [p_z] = x_{\ell+1},$$

where p_j is a rule (II), $1 \leq j \leq z$, for some $z \geq 0$. Every rule $p_j \in P'$ was introduced based on some rule $q_j \in P_{i_{m+1}}$, for some $1 \leq i_{m+1} \leq n$. Then, there exists a derivation

$$v = x'_0 \Rightarrow x'_1\ [q_1] \Rightarrow x'_2\ [q_2] \Rightarrow \cdots \Rightarrow x'_{z-1}\ [q_{z-1}] \Rightarrow x'_z\ [q_z] = w,$$

where $\gamma(w') = w$. Moreover, by Claim 22, it can be expressed as $v \Rightarrow^t_{i_{m+1}} w$ in G and the claim holds.

b) If $d_{\ell+1} = III$, by the rules (III) applied in $x_\ell \Rightarrow^{d_{\ell+1}} x_{\ell+1}$, with $v = w$, $\gamma(w') = w$ holds trivially, which completes the proof. $\square$

By Claims 21 through 23, $S \Rightarrow^t_{i_1} \cdots \Rightarrow^t_{i_m} w$ in G, for some $m \geq 0$, $1 \leq i_j \leq n$, $1 \leq j \leq m$, if and only if $S \Rightarrow^* w'$ in G', where $\gamma(w') = w$. If $w \in T^*$, $\gamma(w') = w' = w$. Then, $\mathrm{L}(G^t) = \mathrm{L}(G')$ and Theorem 6.4.1 holds. $\square$

Theorem 6.4.2. *A language L is context-free iff there is a constant $k \geq 0$ and a cooperating distributed grammar system $G = (V, T, S, P_1, \ldots, P_n)$ of degree $n \geq 1$ such that $L = \mathrm{L}(G^{\geq h})$, for any $h \geq 1$, and for every $w \in \mathrm{L}(G^{\geq h})$, there exists a derivation d*

$$d = S \Rightarrow^{\geq h}_{i_1} \cdots \Rightarrow^{\geq h}_{i_m} w$$

in G of a length $m \geq 1$, where $1 \leq i_j \leq n$, $1 \leq j \leq m$, with ${}_G\triangle(d)$ in which there exists k or fewer component-change cuts.

Proof. Construction. Let $h, k \geq 0$ be two constants. Let $G = (V, T, S, P_1, \ldots, P_n)$ be a CDGS of degree $n \geq 1$ such that for every $w \in \mathrm{L}(G^{\geq h})$, there exists a derivation d

$$d = S \Rightarrow^{\geq h}_{i_1} \cdots \Rightarrow^{\geq h}_{i_m} w$$

in G of a length $m \geq 1$, where $1 \leq i_j \leq n$, $1 \leq j \leq m$, with ${}_G\triangle(d)$ in which there exists k or fewer component-change cuts. Define CFG $G' = (V', T, P', S')$ as follows. Initially, set

$$N' = \{\langle A, x, y\rangle \mid A \in N, x \in \{1, 2, \ldots, n, \varepsilon\}^k, y \in \{1^h, 2^h, \ldots, n^h, \varepsilon\}^k\} \cup \{S'\}$$

and $P' = \emptyset$. Define the function $\tau \colon \mathbb{N} \times \mathbb{N}^* \to \mathbb{N}^*$ as $\tau(l, a_1a_2 \cdots a_j) = a_l a_{2\cdot l} \cdots a_{c\cdot l}$, where $c \cdot l \leq j$, $(c+1) \cdot l > j$, for $l, c, a_i \in \mathbb{N}$, $1 \leq i \leq j$. Define the morphism $\gamma \colon N' \cup T \to N \cup T$ as $\gamma(a) = a$, for $a \in T$, $\gamma(\langle A, x, y\rangle) = A$, for $\langle A, x, y\rangle \in N'$. Construct P' by performing (I) through (V) given next.

(I) For all $x \in \{1^h, 2^h, \ldots, n^h, \varepsilon\}^k$, $x \neq \varepsilon$, add

$$S' \to \langle S, \tau(h, x), x\rangle \text{ to } P';$$

(II) for all $i \in \{1, \ldots, n\}$, $A \to w \in P_i$, $x \in \{1, 2, \ldots, n, \varepsilon\}^{k-1}$, and $y \in \{1, 2, \ldots, n, \varepsilon\}^{(h\cdot k)-1}$, where $w = w_0A_1w_2A_2w_3 \cdots w_{l-1}A_lw_l$, $w_0, w_j \in T^*$, $A_j \in N$, $1 \leq j \leq l$, for some $l \geq 1$, add

$$\langle A, ix, iy\rangle \to w_0\langle A_1, ix, y_1\rangle w_2\langle A_2, ix, y_2\rangle w_3 \cdots w_{l-1}\langle A_l, ix, y_l\rangle w_l$$

to P', where $y = \mathrm{perm}(y_1y_2 \cdots y_l)$ such that for every $a_q \in \{1, \ldots, n\}$, where $y_c = a_1a_2 \cdots a_q \cdots a_p$, $1 \leq c \leq l$, $1 \leq q \leq p$, $p \geq 0$, y is of the form

$$y = u_0a_1u_1a_2u_2 \cdots u_{q-1}a_qu_q \cdots u_{p-1}a_pu_p;$$

(III) for all $i, j \in \{1, \ldots, n\}$, $A \to w \in P_i$, $x \in \{1, 2, \ldots, n, \varepsilon\}^{k-1}$, and $y \in \{1, 2, \ldots, n, \varepsilon\}^{(h\cdot k)-1}$, where $i \neq j$, $w = w_0A_1w_2A_2w_3 \cdots w_{l-1}A_lw_l$, $w_0, w_t \in T^*$, $A_t \in N$, $1 \leq t \leq l$, for some $l \geq 1$, add

$$\langle A, ix, jy\rangle \to w_0\langle A_1, ix, y_1\rangle w_2\langle A_2, ix, y_2\rangle w_3 \cdots w_{l-1}\langle A_l, ix, y_l\rangle w_l$$

to P', where $jy = \mathrm{perm}(y_1y_2 \cdots y_l)$ such that for every $a_q \in \{1, \ldots, n\}$, where $y_c = a_1a_2 \cdots a_q \cdots a_p$, $1 \leq c \leq l$, $1 \leq q \leq p$, $p \geq 0$, jy is of the form

$$jy = u_0a_1u_1a_2u_2 \cdots u_{q-1}a_qu_q \cdots u_{p-1}a_pu_p;$$

(IV) for all $A \to w \in P_i$ and $x \in \{1, 2, \dots, n, \varepsilon\}^k$, where $w \in T^*$, add

$$\langle A, x, \varepsilon \rangle \to w \text{ to } P'.$$

(V) for all $i, j \in \{1, \dots, n\}$, where $i \neq j$, $A \in N$, $x \in \{1, 2, \dots, n, \varepsilon\}^{k-1}$, and $y \in \{1, 2, \dots, n, \varepsilon\}^{(h \cdot k)-1}$, add

$$\langle A, ix, jy \rangle \to \langle A, x, jy \rangle \text{ to } P';$$

Basic Idea. Every nonterminal $\langle A, x, y \rangle \in N'$ encodes the corresponding nonterminal $A \in N$, a string of following simulated active components of G, x, and a string of h-tuples of components corresponding to the rules to be applied, y. By an initial rule (I), all the following component activations are nondeterministically planed. The leftmost symbol of x denotes the current active component. When a component i is active, the leftmost is in y are continuously consumed with every application of a rule (II) simulating a rule in P_i. After consuming of h is, additional rules from P_i may be simulated by the rules (III), so, $\geq h$ mode is followed. In both (II) and (III), the rules to be applied, encoded in y, are distributed into the right-hand-side nonterminals while preserving their mutual order. Then, if $y = \varepsilon$ and, thus, the nonterminal does not encode any rules to be applied, it can be rewritten to a terminal string by some rule (IV). Finally, by the rules (V), a component changes are simulated.

Claim 24. For every derivation

$$S \Rightarrow_{i_1}^{\geq h} \cdots \Rightarrow_{i_m}^{\geq h} w$$

in G of a length $m \geq 0$, for $1 \leq i_j \leq n$, $1 \leq j \leq m$, there exists a derivation $S \Rightarrow^* w'$ in G', where $\gamma(w') = w$.

We prove Claim 24 by induction on m.

Proof. Basis. If $m = 0$, $S = w$. Then, by some rule (I), $S' \Rightarrow \langle S, x, y \rangle$, where $\gamma(\langle S, x, y \rangle) = S$, and the claim holds.

Induction Hypothesis. Suppose there exists ℓ such that the claim holds for all $0 \leq m \leq \ell$.

Induction Step. Consider a derivation $S \Rightarrow_{i_1}^{\geq h} \cdots \Rightarrow_{i_{\ell+1}}^{\geq h} w$ in G, for some $1 \leq i_{\ell+1} \leq n$. Since $\ell + 1 \geq 1$, we can express the derivation as

$$S \Rightarrow_{i_1}^{\geq h} \cdots \Rightarrow_{i_\ell}^{\geq h} v \Rightarrow_{i_{\ell+1}}^{\geq h} w.$$

By the induction hypothesis $S \Rightarrow^* v'$ in G', where $\gamma(v') = v$. Then, v is of the form $u_0 A_1 u_1 \cdots A_z u_z$, $u_0, u_j \in T^*$, $A_j \in N$, $1 \leq j \leq z$, and v' is of the form

$$u_0 \langle A_1, ix, {i_{\ell+1}}^{h_1} y_1 \rangle u_1 \cdots \langle A_z, ix, {i_{\ell+1}}^{h_z} y_z \rangle u_z,$$

where $h_1 + h_2 + \cdots + h_z = h$, $x = \tau(h, y)$, and $y = \operatorname{perm}(y_1 y_2 \cdots y_z)$ such that for every $a_q \in \{1, \dots, n\}$, where $y_c = a_1 a_2 \cdots a_q \cdots a_p$, $1 \leq c \leq z$, $1 \leq q \leq p$, $p \geq 0$, y is of the form $y = u_0 a_1 u_1 a_2 u_2 \cdots u_{q-1} a_q u_q \cdots u_{p-1} a_p u_p$, for some $1 \leq i \leq n$. Without any loss of generality, suppose that the string x corresponds to the following active components of G. If $i \neq i_{\ell+1}$, $x = i_{\ell+1} x'$ and by the rules (V)

$$u_0\langle A_1, ix, {i_{\ell+1}}^{h_1}y_1\rangle u_1 \cdots \langle A_z, ix, {i_{\ell+1}}^{h_z}y_z\rangle u_z \Rightarrow^z$$
$$u_0\langle A_1, i_{\ell+1}x', {i_{\ell+1}}^{h_1}y_1\rangle u_1 \cdots \langle A_z, i_{\ell+1}x', {i_{\ell+1}}^{h_z}y_z\rangle u_z;$$

otherwise, $x' = x$. Express $v \Rightarrow^{\geq h}_{i_{\ell+1}} w$ as

$$v = w_0 \Rightarrow w_1\ [p_1] \Rightarrow w_2\ [p_2] \Rightarrow \cdots \Rightarrow w_o\ [p_o] = w,$$

where $p_j \in P_{i_{\ell+1}}$, for $1 \leq j \leq o$, $o \geq h$. Suppose there exists w_j, $1 \leq j \leq o-1$, where $S \Rightarrow^* w'_j$ in G', $\gamma(w'_j) = w_j$, and

$$w_j = w_{j_1}Aw_{j_2} \Rightarrow w_{j_1}b_0B_1b_1 \cdots B_rb_rw_{j_2}\ [p_{j+1}] = w_{j+1}$$

by some rule $p_{j+1}\colon A \to b_0B_1b_1 \cdots B_rb_r \in P_{i_{\ell+1}}$, where $b_0, b_l \in T^*$, $B_l \in N$, $1 \leq l \leq r$, for some $r \geq 0$. Then, $w'_j = w'_{j_1}\langle A, i_{\ell+1}x', y'\rangle w'_{j_2}$, and by the construction of G' there exist the following three cases depending on y' and r.

1. Suppose $y' = i_{\ell+1}\bar{y}$ and $r > 0$, for any $\bar{y} \in \{1, 2, \ldots, n, \varepsilon\}^*$. Then, there exists a rule (II),

   $$\langle A, i_{\ell+1}x', i_{\ell+1}\bar{y}\rangle \to b_0\langle B_1, i_{\ell+1}x', \bar{y}_1\rangle b_2 \cdots \langle B_r, i_{\ell+1}x', \bar{y}_r\rangle b_r \in P'$$

   by which

   $$w_{j_1}\langle A, i_{\ell+1}x', i_{\ell+1}\bar{y}\rangle w_{j_2} \Rightarrow w_{j_1}b_0\langle B_1, i_{\ell+1}x', \bar{y}_1\rangle b_2 \cdots \langle B_r, i_{\ell+1}x', \bar{y}_r\rangle b_rw_{j_2} = w'_{j+1},$$

   where $\bar{y} = \text{perm}(\bar{y}_1\bar{y}_2 \cdots \bar{y}_r)$ such that for every $a_q \in \{1, \ldots, n\}$, $\bar{y}_c = a_1a_2 \cdots a_q \cdots a_p$, $1 \leq c \leq l$, $1 \leq q \leq p$, $p \geq 0$, $\bar{y}$ is of the form

   $$\bar{y} = u_0a_1u_1a_2u_2 \cdots u_{q-1}a_qu_q \cdots u_{p-1}a_pu_p.$$

   Obviously, $\gamma(w'_{j+1}) = w_{j+1}$.

2. Suppose $y' = \bar{i}\bar{y}$ and $r > 0$, for any $\bar{i} \in \{1, 2, \ldots, n\}$ and $\bar{y} \in \{1, 2, \ldots, n, \varepsilon\}^*$, where $\bar{i} \neq i_{\ell+1}$. Then, there exists a rule (III),

   $$\langle A, i_{\ell+1}x', \bar{i}\bar{y}\rangle \to b_0\langle B_1, i_{\ell+1}x', \bar{y}_1\rangle b_2 \cdots \langle B_r, i_{\ell+1}x', \bar{y}_r\rangle b_r \in P'$$

   by which

   $$w_{j_1}\langle A, i_{\ell+1}x', i_{\ell+1}\bar{y}\rangle w_{j_2} \Rightarrow w_{j_1}b_0\langle B_1, i_{\ell+1}x', \bar{i}\bar{y}_1\rangle b_2 \cdots \langle B_r, i_{\ell+1}x', \bar{i}\bar{y}_r\rangle b_rw_{j_2} = w'_{j+1},$$

   where $\bar{i}\bar{y} = \text{perm}(\bar{y}_1\bar{y}_2 \cdots \bar{y}_r)$ such that for every $a_q \in \{1, \ldots, n\}$, $\bar{y}_c = a_1a_2 \cdots a_q \cdots a_p$, $1 \leq c \leq l$, $1 \leq q \leq p$, $p \geq 0$, $\bar{y}$ is of the form

   $$\bar{y} = u_0a_1u_1a_2u_2 \cdots u_{q-1}a_qu_q \cdots u_{p-1}a_pu_p.$$

   Obviously, $\gamma(w'_{j+1}) = w_{j+1}$.

3. Suppose $y' = \varepsilon$ and $r = 0$. Then, $w_{j+1} = w_{j_1}b_0w_{j_2}$ and there exists a rule (IV), $\langle A, i_{\ell+1}x', \varepsilon\rangle \to b_0$, by which

   $$w_{j_1}\langle A, i_{\ell+1}x', \varepsilon\rangle w_{j_2} \Rightarrow w_{j_1}b_0w_{j_2} = w'_{j+1},$$

   where $\gamma(w'_{j+1}) = w_{j+1}$.

Notice that for $w_0 = w_j$, $S \Rightarrow^* w'_j$ holds trivially by the induction basis which completes the proof. □

Claim 25. Every derivation of $w \in \mathrm{L}(G')$ can be derived in G' as follows.

$$S' = x_0 \Rightarrow^{d_1} x_1 \Rightarrow^{d_2} x_2 \Rightarrow^{d_3} \cdots \Rightarrow^{d_1} x_{h+1} \Rightarrow^{d_h} x_h = w,$$

for some $h \geq 0$, where $d_i \in \{I, II|III|IV, V\}$, $1 \leq i \leq h$, so that

1. *if $d_i = I$, then $x_{i-1} = S'$, $x_i = \langle S, x, y \rangle$, for some $x \in \{1, 2, \dots, n, \varepsilon\}^k$ and $y \in \{1^h, 2^h, \dots, n^h, \varepsilon\}^k$, and by some rule (I), $S' \Rightarrow \langle S, x, y \rangle$.*
2. *if $d_i = II|III|IV$, then*

$$x_{i-1} = u_0\langle A_1, jx, j^{h_1}y_1\rangle u_1\langle A_2, jx, j^{h_2}y_2\rangle u_2 \cdots u_{z-1}\langle A_z, j^{h_z}x, y_z\rangle u_z,$$

where $u_0, u_l \in T^*, \langle A_l, jx, y_l\rangle \in N'$, $1 \leq j \leq n$, $x \in \{1, 2, \dots, n, \varepsilon\}^{k-1}$, $y_l \in \{1, 2, \dots, n, \varepsilon\}^*$, $h_1 + h_2 + \dots + h_z = h$, $1 \leq l \leq z$, and $z \geq 0$, and $x_{i-1} \Rightarrow^* x_i$ only by the rules (II), (III), and (IV) so that

$$x_i = u'_0\langle B_1, jx, y'_1\rangle u'_1\langle B_2, jx, y'_2\rangle u'_2 \cdots u'_{z'-1}\langle B_{z'}, jx, y'_{z'}\rangle u'_{z'},$$

where $u'_0, u'_{l'} \in T^*, \langle B_{l'}, jx, y'_{l'}\rangle \in N'$, $1 \leq l' \leq z'$, for some $z' \geq 0$, and $y_1y_2\cdots y_z = \mathrm{perm}(y'_1y'_2\cdots y'_{z'})$.

3. *if $d_i = V$, then*

$$x_{i-1} = u_0\langle A_1, jx, y_1\rangle u_1\langle A_2, jx, y_2\rangle u_2 \cdots u_{z-1}\langle A_z, jx, y_z\rangle u_z,$$

where $u_0, u_l \in T^*, \langle A_l, jx, y_l\rangle \in N'$, $1 \leq j \leq n$, $x, y_l \in \{1, 2, \dots, n, \varepsilon\}^*$, $1 \leq l \leq z$, $z \geq 0$, and $ju \notin \{y_1, y_2, \dots, y_z\}$, for any $u \in \{1, 2, \dots, n, \varepsilon\}^*$, and $x_{i-1} \Rightarrow^* x_i$ only by the rules (V) of the form $\langle A, jx, y\rangle \to \langle A, x, y\rangle$, where

$$x_i = u_0\langle A_1, x, y_1\rangle u_1\langle A_2, x, y_2\rangle u_2 \cdots u_{z-1}\langle A_z, x, y_z\rangle u_z.$$

Proof. Let us inspect the rules of G'. Since S' is the start symbol and never occurs on the right-hand side of any rule, 1. is trivial. Observe (II)-(V) of the construction; the rules are designed that the left hand sides of (II)-(IV) and (V) are mutually exclusive. In (II)-(IV), while rewriting $\langle A, jx, y\rangle$ and, thus, simulating activated component P_j of G, each right-hand-side nonterminal inherits ix. If $y = jy'$, only the rules (II) are applicable. Observe

$$x_{i-1} = u_0\langle A_1, jx, j^{h_1}y_1\rangle u_1\langle A_2, jx, j^{h_2}y_2\rangle u_2 \cdots u_{z-1}\langle A_z, j^{h_z}x, y_z\rangle u_z$$

in 2. Since $h_1 + h_2 + \dots + h_z = h$, there must be precisely h applications of the rules (II). Additionally, G' may apply possibly any number of the rules from (III) and (IV) which follows $\geq h$ mode. Since G' is a CFG, without any loss of generality, we can reorder any derivation to first apply all rules (II)-(IV) before applying any rule (V) which completes 2. Finally, after simulating $\geq h$ derivation steps of P_j all nonterminals are of the form $\langle A, ix, y\rangle$ with same i and x, where i is not a prefix of y. Then, all nonterminals can be rewritten by the rules (V) which completes 3. The fully detailed rigorous proof is left to the reader. □

Claim 26. For every derivation

$$S' = x_0 \Rightarrow^{d_1} x_1 \Rightarrow^{d_2} x_2 \Rightarrow^{d_3} \cdots \Rightarrow^{d_{h-1}} x_{r-1} \Rightarrow^{d_h} x_r = w',$$

in G', for some $r \geq 1$, where d_i satisfies Claim 25, and $1 \leq i \leq r$, there is a derivation

$$S \Rightarrow_{i_1}^{\geq h} \cdots \Rightarrow_{i_m}^{\geq h} w$$

in G of a length $m \geq 0$, for $1 \leq i_j \leq n$, $1 \leq j \leq m$, where $\gamma(w') = w$.

We proof Claim 23 by induction on r.

Proof. Basis. If $r = 1$, $S' \Rightarrow^I \langle S, x, y \rangle = w'$, for some $x \in \{1, 2, \ldots, n, \varepsilon\}^k$ and $y \in \{1^h, 2^h, \ldots, n^h \varepsilon\}^k$. Then, with $S = w$, $\gamma(w') = w$, so the basis holds.

Induction Hypothesis. Suppose there exists ℓ such that the claim holds for all $1 \leq r \leq \ell$.

Induction Step. Consider a derivation

$$S' \Rightarrow^{d_1} \cdots \Rightarrow^{d_{\ell+1}} x_{\ell+1} = w',$$

in G'. Since $\ell + 1 \geq 2$, there exists $v' = x_\ell$, where

$$S' \Rightarrow^{d_1} \cdots \Rightarrow^{d_\ell} x_\ell \Rightarrow^{d_{\ell+1}} x_{\ell+1}.$$

By the induction hypothesis, $S \Rightarrow_{i_1}^{\geq h} \cdots \Rightarrow_{i_m}^{\geq h} v$, where $\gamma(v') = v$, for some $m \geq 0$. Next, we consider two cases depending on $d_{\ell+1}$.

a) If $d_{\ell+1} = II|III|IV$, $x_\ell \Rightarrow^{d_{\ell+1}} x_{\ell+1}$ can be expressed as

$$x_\ell = x_0 \Rightarrow x_1\ [p_1] \Rightarrow x_2\ [p_2] \Rightarrow \cdots \Rightarrow x_{z-1}\ [p_{z-1}] \Rightarrow x_z\ [p_z] = x_{\ell+1},$$

where p_j is a rule (II), (III), or (IV), $1 \leq j \leq z$, for some $z \geq 0$. Every rule $p_j \in P'$ was introduced based on some rule $q_j \in P_{i_{m+1}}$, for some $1 \leq i_{m+1} \leq n$. Then, there exists a derivation

$$v = x'_0 \Rightarrow x'_1\ [q_1] \Rightarrow x'_2\ [q_2] \Rightarrow \cdots \Rightarrow x'_{z-1}\ [q_{z-1}] \Rightarrow x'_z\ [q_z] = w,$$

where $\gamma(w') = w$. Moreover, by Claim 25, it can be expressed as $v \Rightarrow_{i_{m+1}}^{\geq h} w$ in G and the claim holds.

b) If $d_{\ell+1} = V$, by the rules (V) applied in $x_\ell \Rightarrow^{d_{\ell+1}} x_{\ell+1}$, with $v = w$, $\gamma(w') = w$ holds trivially, which completes the proof. □

By Claims 24 through 26, $S \Rightarrow_{i_1}^{\geq h} \cdots \Rightarrow_{i_m}^{\geq h} w$ in G, for some $m \geq 0$, $1 \leq i_j \leq n$, $1 \leq j \leq m$, if and only if $S \Rightarrow^* w'$ in G', where $\gamma(w') = w$. If $w \in T^*$, $\gamma(w') = w' = w$. Then, $\mathrm{L}(G^{\geq h}) = \mathrm{L}(G')$ and Theorem 6.4.2 holds. □

Theorem 6.4.3. *A language L is context-free iff there is a constant $k \geq 0$ and a cooperating distributed grammar system $G = (V, T, S, P_1, \ldots, P_n)$ of degree $n \geq 1$ such that $L = \mathrm{L}(G^{=h})$, for any $h \geq 1$, and for every $w \in \mathrm{L}(G^{=h})$, there exists a derivation d*

$$d = S \Rightarrow_{i_1}^{=h} \cdots \Rightarrow_{i_m}^{=h} w$$

in G of a length $m \geq 1$, where $1 \leq i_j \leq n$, $1 \leq j \leq m$, with ${}_G\triangle(d)$ in which there exists k or fewer component-change cuts.

Proof. Prove the theorem by analogy with the proof of Theorem 6.4.2 omitting step (III) of the construction. □

As a special case, let us consider the following theorem for $= h$ derivation mode, where, however, we force the component changes to never chose the same component twice in a row.

Theorem 6.4.4. *A language L is finite iff there is a constant $k \geq 0$ and a cooperating distributed grammar system $G = (V, T, S, P_1, \ldots, P_n)$ of degree $n \geq 1$ such that $L = \mathrm{L}(G^{=h})$ and for every $w \in \mathrm{L}(G^{=h})$, there exists a derivation d*

$$d = S \Rightarrow_{i_1}^{=h} \cdots \Rightarrow_{i_m}^{=h} w$$

in G of a length $m \geq 1$, where $1 \leq i_j \leq n$, $1 \leq i_m \leq n$, $1 \leq j < m$, and $i_j \neq i_{j+1}$, with ${}_G\triangle(d)$ in which there exists k or fewer component-change cuts.

Proof. Let $h, k \geq 1$ be two constants. Let $G = (V, T, S, P_1, \ldots, P_n)$ be a CDGS of degree $n \geq 1$ such that $L = \mathrm{L}(G^{=h})$, for any $h \geq 1$, and for every $w \in \mathrm{L}(G^{=h})$, there exists a derivation d

$$d = S \Rightarrow_{i_1}^{=h} \cdots \Rightarrow_{i_m}^{=h} w$$

in G of a length $m \geq 1$, where $1 \leq i_j \leq n$, $1 \leq j \leq m$ with ${}_G\triangle(d)$ in which there exists k or fewer component-change cuts. Then, $m \leq h \cdot (k+1)$ and since $P_1 \cup P_2 \cup \cdots P_n$ is finite, there exist only a finite number of possible derivations. Therefore, G generates a finite number of terminal strings. □

In the following chapter, we demonstrate how to use the achieved results to obtain a positive prove of context-freeness of a language.

Chapter 7

How to Prove Context-Freeness

Even though this thesis is mainly theoretically oriented, in the present chapter, however, we demonstrate its practical impact. As presented in Section 4.1, a proof of context-freeness is not a straightforward process. Indeed, unlike Workspace Theorem for context-sensitive languages, the theory of formal languages lacked a proof scheme to more automate the process of proving that a certain language is context-free. Nevertheless, in Chapter 6 we described necessary but also sufficient conditions for language to be context-free based on general grammars, scattered context grammars, and cooperating distributed grammar systems. Under the given derivation-tree restrictions these grammars characterize precisely the family of context-free languages. Moreover, we can obtain a positive proof of a context-freeness of a language L by following the next three-step proof scheme.

1. Construct a general or scattered context grammar in binary form, or cooperating distributed grammar system G.
2. Prove $\mathrm{L}(G) = L$.
3. Prove that G satisfies conditions given by the respective theorem.

As a result $L \in \mathbf{CF}$. The following three examples show how to use this proof scheme to prove context-freeness of some non-trivial context-free languages.

Example 7.0.1. Reconsider the grammar G from Example 3.2.3. Following the proof scheme sketched above, we next prove that $\mathrm{L}(G) \in \mathbf{CF}$.

Consider G constructed in Example 3.2.3. Next, we show that for G,

$$\mathrm{L}(G) = \{w \in (A \cup \{\varepsilon\})(BA)^*(B \cup \{\varepsilon\}) \mid \#_a(w) = \#_b(w),$$
$$A = \{a^i \mid 1 \leq i \leq 5\}, B = \{b^i \mid i \geq 3\}, \text{ and } |w| > 0\}.$$

Without any loss of generality, every terminal derivation of G can be divided into the following 5 phases, where each rule may be used only in a specific phase:

(a) (1)–(4) (b) (5)–(11) (c) (12)–(17) (d) (18)–(31) (e) (32)–(33)

Next, we describe these phases in a greater detail.

(a) First, we generate one of the following two strings by rules (1) through (4).

$$Z_a X B_x, Z_b X A_x$$

Possibly applicable rule (25) may be postponed for phase (d) without affecting the derivation, since rules in the previous phases cannot rewrite A_x.

(b) The rules (5) through (11) are the only with X, X_a, or X_b on their left-hand sides, therefore, we can group all their applications in a sequence to get a sentential form from

$$\{Z_a, Z_b\}\{A, B\}^*\{A_x, B_x\}.$$

(c) The rules (12) through (17) possibly shift Z_a or Z_b to the right and rewrite it to A or B, respectively. Since these rules are the only with Z_a, Z_b on their left-hand sides, they can be always prioritized before the rest of rules without any loss of generality.

$$\{A, B\}^*\{A_x, B_x\}$$

(d) All the remaining rules may be applied in this phase. However, we can exclude rules (32) and (33), so we get a sentential form from

$$\{\overline{a}, \overline{b}\}^*.$$

(e) Since rules (32) and (33) are context-free and produce terminal symbols, they can be always postponed until the end of any successful derivation.

$$\{a, b\}^* = T^*$$

Let us add a few remarks concerning (a) through (e).

Phase (a) is very straightforward. Only notice that it is decided whether the generated string finally ends with a or b and the paired symbol is stored in Z_a or Z_b for phase (c).

In phase (b) an arbitrary string of As and Bs is generated from the initial symbol X. However, for every A, one B is generated and vice versa, so their numbers are always kept equal.

In phase (a) the grammar decides about the last symbol and stores the paired one, which, however, need not to be the first one. Therefore, phase (c) determines its final position, while possibly shifting it to the right and finally rewriting to A or B.

Phase (d) is the most tricky. It starts with a sentential form wc, where $w \in \{A, B\}^*$, $c \in \{A_x, B_x\}$. Informally speaking, it consists of the sequences of As which should be at most 5 symbols long, and Bs which should be at least 3 symbols long. Rules (18) through (31) are designed to ensure these restrictions. To give an example, suppose wc is as follows.

$$wc = AAAABBBBABBBAA_x$$

First, by rules (18) through (20) the last symbol in every sequence is marked with index x. Otherwise, rules (24) through (28) and rule (31) never become applicable and all the unmarked sequences become permanent resulting into an unsuccessful derivation. The last sequence is already marked.

$$\begin{aligned}
& AAAABBBBABBBAA_x \\
\Rightarrow\ & AAAA_xBBBBABBBAA_x && [(18)] \\
\Rightarrow\ & AAAA_xBBBBA_xBBBAA_x && [(18)] \\
\Rightarrow\ & AAAA_xBBBB_xA_xBBBAA_x && [(20)] \\
\Rightarrow\ & AAAA_xBBBB_xA_xBBB_xAA_x && [(19)]
\end{aligned}$$

Notice, one symbol sequence of As is legal. Then, every sequence of As is processed in left-to-right direction by rules (21) through (24), but can be successfully rewritten earlier by rules (25) through (28), in the case it consists of less than 5 symbols. Thus, a longer sequence leads to an unsuccessful derivation.

$$\begin{array}{lll} & AAAA_xBBBB_xA_xBBB_xAA_x & \\ \Rightarrow & \overline{a}1AA_xBBBB_xA_xBBB_xAA_x & [(21)] \\ \Rightarrow & \overline{aa}2A_xBBBB_xA_xBBB_xAA_x & [(22)] \\ \Rightarrow & \overline{aaaa}BBBB_xA_xBBB_xAA_x & [(27)] \\ \Rightarrow & \overline{aaaa}BBBB_x\overline{a}BBB_xAA_x & [(25)] \\ \Rightarrow & \overline{aaaa}BBBB_x\overline{a}BBB_x\overline{aa} & [(26)] \end{array}$$

If the processing does not start from the leftmost symbol in the current sequence, it remains permanent. Every sequence of Bs is processed by applying rule (29), zero or multiple times rule (30), and finally rule (31). It ensures the lengths of sequences of Bs are at least 3 symbols.

$$\begin{array}{lll} & \overline{aaaa}BBBB_x\overline{a}BBB_x\overline{aa} & \\ \Rightarrow & \overline{aaaab}B_xBB_x\overline{a}BBB_x\overline{aa} & [(29)] \\ \Rightarrow & \overline{aaaabb}B_xB_x\overline{a}BBB_x\overline{aa} & [(30)] \\ \Rightarrow & \overline{aaaabbbba}BBB_x\overline{aa} & [(31)] \\ \Rightarrow & \overline{aaaabbbbab}B_xB_x\overline{aa} & [(29)] \\ \Rightarrow & \overline{aaaabbbbabbbaa} & [(31)] \end{array}$$

Notice, it depends on the order of applied rules only within one sequence. Multiple sequences may be processed at random without affecting the derivation.

In phase (e), a resulting terminal string is generated by rules (32) and (33).

$$\overline{aaaabbbbabbbaa} \Rightarrow^* aaaabbbbabbbaa$$

Therefore, if the derivation is terminating, we achieve a string with an equal number of as and bs, where every sequence of as is at most 5 symbols long and every sequence of bs is at least 3 symbols long.

Grammar G is obviously a monotone general grammar in the binary form. Let us now show that for any $x \in \mathrm{L}(G)$, there is ${}_G\triangle_x \in {}_G\blacktriangle$, where any two neighbouring paths contain no more than 2 pairs of context-dependent nodes.

Every pair of context-dependent nodes in ${}_G\triangle_x$ corresponds to one non-context-free rule in $S \Rightarrow^* x$. Consider the six phases sketched above. Observe that phases (a), (b), and (e) contain only context-free rules, so we have only to investigate (c) and (d). On the other hand, (c) and (d) contain no rule of the form $A \to BC$, thus the number of neighbouring paths remains unchanged.

In (c) by rules (12) through (17) the derivation may proceed in left-to-right direction through the whole sentence form (except the rightmost symbol) introducing a context dependency between every pair of neighbouring paths.

In (d), first, the context dependency is introduced between all neighbouring paths representing the borders between the sequences of As and Bs by rules (18) through (20). Second, every sequence of As or Bs is processed in the left-to-right direction by non-context-free rules (21) through (31) introducing a context dependency between all neighbouring paths representing symbols inside the sequences of As and Bs.

No other non-context-free rule is applied, therefore, no other context-dependent pair of nodes can occur. Then, every pair of neighbouring paths may contain at most one context-dependent pair of nodes introduced in phase (c) and one introduced in phase (d).

Since G is a monotone GG in the binary form, where for every $x \in \mathrm{L}(G)$, there is ${}_G\triangle_x \in {}_G\blacktriangle$, where any two neighbouring paths contain no more than 2 pairs of context-dependent nodes, by Theorem 6.1.2, $\mathrm{L}(G) \in \mathbf{CF}$. □

Example 7.0.2. Let $G = (V, T, P, S)$ be an SCG, where

$$V = \{S, A, B, \bar{S}, \bar{A}, \bar{B}, \bar{a}, \bar{b}, a, b, A_1, A_2, A_3, A_4, A_5, B_1, B_2, B_3, B_4, B_5\},$$

$T = \{a, b\}$, and

$$
\begin{array}{llll}
P = \{ & 1\colon (S) \to (\bar{S}S), & 4\colon (A) \to (\bar{a}\bar{A}), & 8\colon (A, B) \to (A_1, B_1), \\
& 2\colon (\bar{S}) \to (AB), & 5\colon (\bar{A}) \to (A\bar{a}), & 9\colon (A, B) \to (A_2, B_2), \\
& 3\colon (S) \to (\varepsilon), & 6\colon (B) \to (\bar{b}\bar{B}), & 10\colon (A, B) \to (A_3, B_3), \\
& & 7\colon (\bar{B}) \to (B\bar{b}), & 11\colon (A, B) \to (A_4, B_4), \\
& & & 12\colon (A, B) \to (A_5, B_5), \\
& & & \\
& 13\colon (A_5) \to (\bar{b}A_4), & 18\colon (B_5) \to (\bar{a}B_4), & 23\colon (\bar{a}) \to (a), \\
& 14\colon (A_4) \to (\bar{b}A_3), & 19\colon (B_4) \to (\bar{a}B_3), & 24\colon (\bar{b}) \to (b), \\
& 15\colon (A_3) \to (\bar{b}A_2), & 20\colon (B_3) \to (\bar{a}B_2), & \\
& 16\colon (A_2) \to (\bar{b}A_1), & 21\colon (B_2) \to (\bar{a}B_1), & \\
& 17\colon (A_1) \to (\bar{b}), & 22\colon (B_1) \to (\bar{a})\ \}. &
\end{array}
$$

G is obviously in the binary form. Observe the rules of G. Without any loss of generality, we can reorder the applications of the rules of G to satisfy six-phased generative process as follows.

(1) Initially, rules 1 through 3 generate a sentential form $w_1 \in \{AB\}^*$.

(2) Then, by rules 4 through 7 $w_1 \Rightarrow_G^* w_2$, where $w_2 = X_1 X_2 \cdots X_h$,

$$X_i = \bar{a}^{m_i} A \bar{a}^{m_i} \bar{b}^{n_i} B \bar{b}^{n_i},$$

$m_i, n_i \geq 0$, $1 \leq i \leq h$, for some $h \geq 0$.

(3) The rules 8 through 12—all the context sensitive rules of G—rewrite As and Bs to their indexed forms. We can assume that they are always applied to the neighbouring A and B; otherwise, there occurs a prefix of a sentential form with more Bs than As which obviously cannot be rewritten to its indexed form. Consequently, every odd index corresponds to the following even one.

$$w_3 = X_1 X_2 \cdots X_h, X_i = \bar{a}^{m_i} A_{g_i} \bar{a}^{m_i} \bar{b}^{n_i} B_{g_i} \bar{b}^{n_i},$$

$m_i, n_i \geq 0$, $1 \leq g_i \leq 5$, $1 \leq i \leq h$, for some $h \geq 0$.

(4) With rules 13 through 17 every indexed A is rewritten to 1 through 5 $\bar{b}$s according to its index.

$$w_4 = X_1 X_2 \cdots X_h, X_i = \bar{a}^{m_i} \bar{b}^{g_i} \bar{a}^{m_i} \bar{b}^{n_i} B_{g_i} \bar{b}^{n_i},$$

$m_i, n_i \geq 0$, $1 \leq g_i \leq 5$, $1 \leq i \leq h$, for some $h \geq 0$.

(5) With rules 18 through 22 every indexed B is rewritten to 1 through 5 $\bar{a}$s according to its index.

$$w_5 = X_1X_2\cdots X_h, X_i = \bar{a}^{m_i}\bar{b}^{g_i}\bar{a}^{m_i}\bar{b}^{n_i}\bar{a}^{g_i}\bar{b}^{n_i},$$

$m_i, n_i \geq 0$, $1 \leq g_i \leq 5$, $1 \leq i \leq h$, for some $h \geq 0$.

(6) Finally, a terminal sentence is generated by rules 23 and 24.

$$w_6 = X_1X_2\cdots X_h, X_i = a^{m_i}b^{g_i}a^{m_i}b^{n_i}a^{g_i}b^{n_i},$$

$m_i, n_i \geq 0$, $1 \leq g_i \leq 5$, $1 \leq i \leq h$, for some $h \geq 0$.

As a result

$$\begin{aligned}\mathrm{L}(G) = \{X_1X_2\cdots X_h \mid X_i = a^{m_i}b^{g_i}a^{m_i}b^{n_i}a^{g_i}b^{n_i},\\ m_i, n_i \geq 0, 1 \leq g_i \leq 5, 1 \leq i \leq h, h \geq 0\}.\end{aligned}$$

To show that $\mathrm{L}(G)$ is context-free, let us observe the phases of the generative process of G in greater detail. First, $S \Rightarrow_G^* w\ [\varrho]$, where $\varrho \in \{1,2,3\}$. Figure 7.0.2 shows the structure of ${}_G\triangle(S \Rightarrow_G^* w\ [\varrho])$.

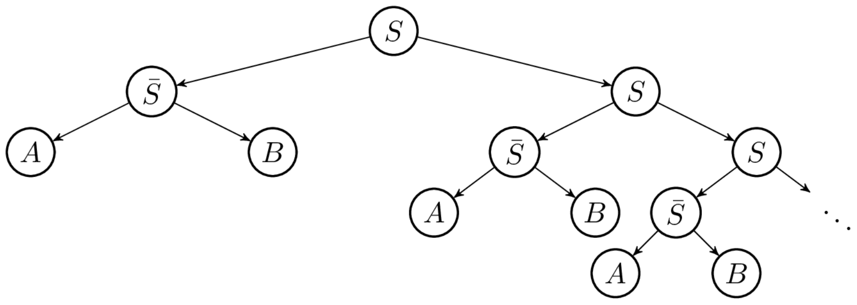


Figure 7.0.1: Graphical representation of ${}_G\triangle_w$

Continuing the derivation, a sentence form is of the form $X_1X_2\cdots X_h$, where X_i corresponds to left subgraph of some S-labelled node in Figure 7.0.2, $1 \leq i \leq h$, $h \geq 0$. Observe that in phase (3) a pair of context sensitive nodes is introduced in every X_i portion of the derivation tree. Since no more context dependencies are introduced we can find a division depicted in Figure 6.

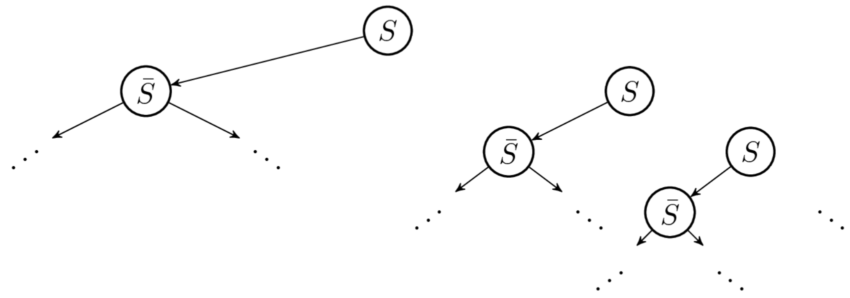


Figure 7.0.2: Division of ${}_G\triangle_w$

The division in Figure 6 satisfies Theorem 6.3.1, for $k = 1$, so, $\mathrm{L}(G) \in \mathbf{CF}$. □

Example 7.0.3. Let $G = (V, T, S, P_1, P_2, P_3, P_4, P_5)$ be a CDGS of degree 5, where

$$V = \{S, A, B, C, \bar{A}, \bar{B}, \bar{C}, X, Y, (,), [,]\},\ T = \{(,), [,]\},\ \text{and}$$

$$\begin{aligned}
P_1 &= \{ \quad 1\colon S \to S,\ 2\colon S \to ABC^{1000},\ 3\colon \bar{A} \to XAY,\ 4\colon \bar{B} \to XBY\ \},\\
P_2 &= \{ \quad 5\colon A \to \bar{A},\ 6\colon C \to \bar{C}\ \},\\
P_3 &= \{ \quad 7\colon B \to \bar{B},\ 8\colon \bar{C} \to \varepsilon\},\\
P_4 &= \{ \quad 9\colon A \to \varepsilon,\ 11\colon B \to \varepsilon\ \},\\
P_5 &= \{ \quad 11\colon X \to (X)X,\ 12\colon X \to \varepsilon\ \},\\
P_6 &= \{ \quad 13\colon Y \to [Y]Y,\ 14\colon Y \to \varepsilon\ \}.
\end{aligned}$$

Consider derivation mode ≥ 2. Observe the rules of G. Every derivation starts with component P_1. Rule 1 is necessary to satisfy the derivation mode condition, then, rule 2 generates the sentence form

$$ABC^{1000}$$

Continuing with component P_4 blocks the derivation in case there is some C remaining. Thus, the derivation continues with component P_2, where both rules 5 and 6 need to be applied not to block the derivation later, rule 6 possibly multiple times—without any loss of generality, suppose always the leftmost C is rewritten—, to generate a sentence form

$$\bar{A}B\bar{C}^kC^l$$

For some $k \geq 1$, $l \geq 0$. The derivation continues with component P_3. The rule 7 must be applied not to block the derivation later. In any successful derivation the number of applications of rule 6 and 8 equals, so, let us suppose that component P_3 always erases all $\bar{C}$s. As a result

$$\bar{A}\bar{B}C^l$$

Next, component P_1 is activated again generating

$$XAYXBYC^l$$

Components P_1, P_2, and P_3 are activated in cycle in the described way always erasing at least one C and generating one X and Y from both A and B.

$$X^nAY^nX^nBY^nC^r$$

After all Cs are removed, P_4 erases A and B.

$$X^nY^nX^nY^n$$

Components P_5 and P_6 are possibly applicable earlier in the derivation, however, since no other component rewrites X and Y, we can postpone them till the very end of every derivation without any loss of generality. First, P_5 generates a string of the well-known Dyck language (see [7]) over $\{(,)\}$ from every X. Second, P_6 generates a string of the Dyck language over $\{[,]\}$ from every Y. Therefore,

$$\begin{aligned}
\mathrm{L}(G) = \{w \in X^nY^nX^nY^n \mid\ & X \text{ and } Y \text{ are Dyck languages}\\
& \text{over } \{(,)\} \text{ and } \{[,]\}, \text{ respectively, and } n \leq 1000\}.
\end{aligned}$$

We prove that $\mathrm{L}(G) \in \mathbf{CF}$ according to Theorem 6.4.2. Since the theorem says that for every $w \in \mathrm{L}(G)$ there exists a derivation satisfying the prescribed restrictions, we have to identify such w which derivation has the highest minimum of necessary component changes.

As we shown previously, every derivation, first, activates components P_1, P_2, and P_3 multiple times in a cycle. However, since every cycle consumes at least one C and there are precisely 1000 Cs present, these components change at most thousand times. Then, component P_4 is activated and the derivation finishes with components P_5 and P_6. The number of possible component changes between components P_5 and P_6 is not limited, nevertheless, the same sentence can be always obtain with only one activation of both and we are looking for a minimum. All together, any $w \in \mathrm{L}(G)$ can be generated with at most 1003 component changes, which corresponds to 1003 component-change cuts in the resulting derivation tree, and, therefore, by Theorem 6.4.2, $\mathrm{L}(G) \in \mathbf{CF}$. □

Chapter 8

Conclusion

Let us conclude this work by summarising all the achieved results and, since it represents brand new area of research, by discussing perspectives for the future investigation.

The present thesis introduced graph-related features of derivation trees of general grammars, regular-controlled grammars, scattered context grammars, and cooperating distributed grammar systems; context dependencies between neighbouring paths, path changes, tree divisions, and component-change cuts, respectively. We placed constant restrictions on the number of these derivation-tree features and proved that they results in context-freeness. This knowledge is on one hand interesting from the theoretical point of view, however, on the other hand, as we demonstrated in the previous chapter, it can be also utilized to obtain a positive prove of context-freeness of a language. Moreover, we introduced simple proof pattern which can simplify some proofs of this kind.

For theoretical computer scientists a question of precise membership of a language in a language family among the Chomsky hierarchy of languages represents essential but challenging task. Direct prove by constructing respective corresponding grammar or automaton requires a unique creative approach and is rarely straightforward. We knew several tools to simplify this kind of proof; namely, pumping lemmas or workspace theorem; however, specifically for indisputably very important family of context-free languages we had no such tool (as we explained in Chapter 4.1). In this thesis we introduced derivation-tree-restriction-based approach to this matter and explained how to obtain a positive proof of context-freeness. This new proof-simplifying tool may help proving context-freeness significantly. Nevertheless, this area of research still offers lot of interesting and challenging questions to be answered in the future. In what follows, we suggests some follow-up topics rising from the subject of this work.

We indisputably selected very important representatives of grammar theory, however, this study is definitely not exhaustive in this sense which rises the following open problem.

Open Problem 1. How to naturally restrict derivation trees of other types of grammars to characterize some language subfamilies? How to use these derivation-tree restrictions while proving that some languages belong (or not) to the language subfamily?

We studied influence of constant restrictions placed on the number of the introduced derivation-tree-related features of generated sentences on the generative power of grammars in question and proved it reduces the generative power significantly. Indeed, a constant restriction is very natural, nevertheless, not the only possible restriction. Certainly, there are several approaches to functional restrictions. Since it is closely related topic indisputably suitable for the following future study, however, it requires thorough investigation, we just give several gists and state some open problems.

First of all, let us consider general grammars and the following open problem.

Open Problem 2. Consider a general grammar G with every $w \in \mathrm{L}(G)$ having a derivation tree with $k|w|$ or less context dependent pairs of nodes within every pair of neighbouring paths. Can we find such a grammar for every language $L \in \mathbf{CS}$? Is there a GG restricted in this way generating a non-context-sensitive language? Or is this restriction in fact an equivalence of Workspace Theorem for context-sensitive languages?

Let us give a rough insight into this matter. It is quite easy to estimate that this less restrictive condition allows the general grammars to generate non-context-free languages as shown by the next example.

Example 8.0.1. Consider a general grammar $G = (\{S, A, B, X\}, \{a, b, c\}, P, S)$ with

$$P = \{S \to AB, A \to aAbX, Xb \to bX, XB \to Bc, A \to \varepsilon, B \to \varepsilon\}.$$

Clearly, $\mathrm{L}(G) = \{a^n b^n c^n \mid n \geq 0\}$ which is the well-known non-context-free context-sensitive language. Moreover, with every a, b, and c introduced into the sentence form, one context dependency between all pairs of symbols between A and B is introduced too. Suppose that in the resulting derivation tree there is a pair of neighbouring paths between which there exists a context dependency for every such triple of symbols; this is the highest possible number of context dependencies among all pairs of neighbouring paths, since no other context dependencies occur. Then, the upper bound of these dependencies is linearly dependent on the length of the resulting sentence, so the conditions stated in Open Problem 2 are met. □

With k related to workspace constant we might be able to simulate this workspace-restricted grammar by context-dependence restricted grammar in above defined way, since the limited workspace results in a limited number of possible configurations. The rest of the proof and even the validity of this idea is, however, beyond the scope of this thesis.

For general grammars, scattered context grammars, and cooperating distributed grammar systems we proved that the constant restrictions on the number of occurrences of introduced derivation-tree-related features result in context-freeness. However, in the case of regular-controlled grammars, this restriction is even stronger. This opens the following questions.

Open Problem 3. Consider non-constant restriction on the number of path-changes of regular-controlled grammars, such as functions over the sentential form lengths. What is the generative power of regular-controlled grammars restricted in this way? How to restrict path changes to obtain versions of regular-controlled grammars characterizing precisely the family of context-free languages?

We also investigate non-context-free derivation-tree-related properties of scattered context grammars and proved that putting constant restrictions on them results in context-freeness. Namely, we showed that if context dependent pairs of nodes are clustered into (unlimited number of) mutually context-independent k-tuples with some given k, the generated language is context-free. Since the original generative power of SCGs covers all recursively enumerable languages this restriction and the resulting loss of generative power is significant which gives rise to the following open problem.

Open Problem 4. Are there any derivation-tree related conditions for scattered context grammars resulting these grammars to characterize precisely some family of languages between context-free and recursively enumerable languages; for example context-sensitive or matrix languages?

An interesting results may be also achieved in this area of research regarding cooperating distributed grammar systems. We introduced the notion of component-change cuts and proved that if we limit this by a constant the resulting language is context-free. However, what if we describe the number of component-changes by some function and, moreover, force the grammar system to perform precisely that many component changes? Let us introduce some initial investigation of this matter, however, the details would require a separate study.

Open Problem 5. Investigate the families of languages generated by CDGSs satisfying the following conditions:

1. Let $f\colon \mathbb{N} \to \mathbb{N}$ be an integer function.
2. Let $\psi \in \{\leq l, =l, \geq l \mid \text{ for some } l \geq 1\} \cup \{t, *\}$.
3. Let $G = (V, T, S, P_1, \dots, P_n)$ be a CDGS of degree $n \geq 1$ such that for every $w \in \mathrm{L}(G^\psi)$, there exists a derivation d

$$d = S \Rightarrow^{\psi}_{i_1} \cdots \Rightarrow^{\psi}_{i_k} w$$

in G, where $1 \leq i_j \leq n$, $1 \leq j \leq k$, so, in ${}_G\triangle(d)$ there is precisely k component-change cuts, where $k = f(x)$, for some $x \in \mathbb{N}$.

Let us denote the language of CDGS G using derivation mode ψ restricted by a function f by $\mathrm{L}(G^{\psi,f})$ and the family of languages of CDGSs of degree n using derivation mode ψ restricted by a function f by $\mathbf{CD}_n^{\psi,f}$.

First, let us state that $\mathbf{CD}_n^{\psi} \subseteq \mathbf{CD}_n^{\psi,f}$ if $f\colon k = x$, for $x \in \mathbb{N}$. Since any number of component-change cuts is allowed, it is, in fact, unrestricted. Observe the following example.

Example 8.0.2. Let

$$G = (\{S, a\}, \{a\}, S, \{S \to Sa, S \to a\})$$

be a CDGS using derivation mode $\Rightarrow^{=1}$. Obviously, G is of degree 1 and uses only regular rules. Moreover, using $\Rightarrow^{=1}$ CDGSs are only as powerful as context-free grammars. However, with $f\colon k = 2^{2^x} - 1$, $x \in \mathbb{N}$,

$$L(G^{=1,f}) = \{a^{2^{2^k}} \mid k \geq 0\} \in (\mathbf{CS} - \mathbf{ET0L}).$$

Notice that the component changes in fact do not change to any different component (there is only one), however, due to the derivation mode, the component is deactivated after every single derivation step, so the component must be activated again. □

We can also consider $f'(x)$, where f' returns the nearest Fibonacci or prime number y, where $y \leq x$, and various combinations to generate languages beyond the ordinary CDGS' families of languages.

Open Problem 6. Categorize CDGS' languages by functional relation between component-change cuts and the length of the generated sentence:

1. Let $f\colon \mathbb{N} \to \mathbb{N}$ be an integer function.
2. Let $\psi \in \{\leq l, = l, \geq l \mid \text{ for some } l \geq 1\} \cup \{t, *\}$.
3. Let $G = (V, T, S, P_1, \ldots, P_n)$ be a CDGS of degree $n \geq 1$ such that for every $w \in L(G^{\psi})$, there exists a derivation d

$$d = S \Rightarrow^{\psi}_{i_1} \cdots \Rightarrow^{\psi}_{i_k} w$$

in G, where $1 \leq i_j \leq n$, $1 \leq j \leq k$, so, in ${}_G\triangle(d)$ there is precisely k component-change cuts, where $k = f(|w|)$.

Observe the following two examples.

Example 8.0.3. Consider a language $L_1 = \{a^{2^n} \mid n \geq 0\}$. $L_1 \in (\mathbf{ET0L} - \mathbf{MT})$. For grammar G_1,

$$G_1 = \{\{S, A, a\}, \{a\}, S, \{S \to AA\}, \{A \to S\}, \{S \to a\}\},$$

and derivation mode $\Rightarrow^t$, $\mathrm{L}(G_1^t) = L_1$. Obviously, for every $w \in \mathrm{L}(G_1^t)$, $k = log_2(|w|)$, where k is the number of component-change cuts. □

Example 8.0.4. Consider a language $L_2 = \{a^n b^n c^n \mid n \geq 0\}$. $L_2 \in (\mathbf{MT} - \mathbf{CF})$. For grammar G_2,

$$G_2 = \{\{S, A, A', B, B', a, b, c\}, \{a, b, c\}, S, \{S \to S, S \to A'B', A \to \varepsilon, B \to \varepsilon\},\\ \{A \to aA'b, B \to B'c\}, \{A' \to A, B' \to B\}\},$$

and derivation mode $\Rightarrow^{\psi}$, where $\psi \in \{=2, t\}$, $\mathrm{L}(G_2^{\psi}) = L_2$. Then, for every $w \in \mathrm{L}(G_2^{\psi})$, $k = |w|$, where k is the number of component-change cuts. □

To generate L_1 we naturally need more powerful language generating model than to generate L_2. However, on the basis of the previous examples, it is evident that while generating sentences of L_1 CDGS performs considerably less component changes. Could we state any general claim on this issue?

References

[1] S. Abraham. Some questions of language theory. In *Proceedings of the 1965 conference on Computational linguistics*, pages 1–11. Association for Computational Linguistics, 1965.

[2] A.V. Aho and J.D. Ullman. *The Theory of Parsing, Translation, and Compiling.* Prentice-Hall, Series in Automatic Computation, 1972.

[3] Y. Bar-Hillel, M. Perles, and E. Shamir. Pumping lemmas for regular sets. *Z. Phonetik. Sprachwiss. Kommunikationsforsch*, 14:143–172, 1961.

[4] G. Cantor. Ueber eine elementare frage der mannigfaltigkeitslehre. *Jahresbericht der Deutschen Mathematiker-Vereinigung*, 14:75–78, 1891.

[5] N. Chomsky. Three models for the description of language. *IRE Transactions on Information Theory*, 2(3):113–124, 1956.

[6] N. Chomsky. On certain formal properties of grammars. *Information and Control*, 2:137–167, 1959.

[7] N. Chomsky and M. P. Schützenberger. The algebraic theory of context-free languages. *Computer programming and formal systems*, pages 118–161, 1963.

[8] D.I.A. Cohen. *Introduction to Computer Theory.* Wiley, New York, 1991.

[9] T.H. Cormen, C.E. Leiserson, and R.L. Rivest. *Introduction to Algorithms.* McGraw-Hill, 2002.

[10] R. Crandall and C. Pomerance. *Prime Numbers: A Computational Perspective.* Springer., 2nd edition, 2005.

[11] E. Csuhaj-Varjú and J. Dassow. On cooperating/distributed grammar systems. *Journal of Information Processing and Cybernetics*, 26:49–63, 1990.

[12] E. Csuhaj-Varjú, J. Kelemen, Gh. Păun, and J. Dassow. *Grammar Systems: A Grammatical Approach to Distribution and Cooperation.* Gordon and Breach Science Publishers, Inc., Newark, NJ, US, 1st edition, 1994.

[13] E. Csuhaj-Varjú and G. Vaszil. Scattered context grammars generate any recursively enumerable language with two nonterminals. *Information Processing Letters*, 110:902–907, 2010.

[14] J. Dassow and G. Păun, editors. *Regulated Rewriting in Formal Language Theory.* Akademie-Verlag, Berlin, 1989.

[15] A. Ehrenfeucht, R. Parikh, and G. Rozenberg. Pumping lemmas for regular sets. *SIAM Journal of Computing*, 10(3):536–541, 1981.

[16] H. Fernau. Scattered context grammars with regulation. *Annals of Bucharest University, Mathematics-Informatics Series*, 45(1):41–49, 1996.

[17] H. Fernau and A. Meduna. A simultaneous reduction of several measures of descriptional complexity in scattered context grammars. *Information Processing Letters*, 86(5):235–240, 2003.

[18] S. Ginsburg, S. A. Greibach, and M. Harrison. One-way stack automata. *Journal of the ACM*, 14(2):389–418, 1967.

[19] S. Ginsburg and E. Spanier. Finite-turn pushdown automata. *SIAM Journal on Control*, 4:429–453, 1968.

[20] S. A. Greibach and J. E. Hopcroft. Scattered context grammars. *Journal of Computer and System Sciences*, 3(3):233–247, 1969.

[21] P. Halmos. *Naive Set Theory.* Springer-Verlag, New York, US, 1974.

[22] J.E. Hopcroft and J.D. Ullman. *Introduction to Automata Theory, Languages, and Computation.* Addison-Wesley, Reading, 1979.

[23] J.M. Howie. *Automata and Languages.* Oxford University Press, Oxford, 1991.

[24] J. Jaffe. A necessary and sufficient pumping lemma for regular languages. *SIGACT News*, pages 48–49, 1978.

[25] N.D. Jones. „a survey of formal language theory“. *Technical Report*, 3, 1966.

[26] N.D. Jones. A note on the index of a context-free language. *Information and Control*, 16:201–202, 1970.

[27] S.-Y. Kuroda. Classes of languages and linear-bounded automata. *Information and Control*, 7:207–223, 1964.

[28] C. Martín-Vide, V. Mitrana, and G. Păun, editors. *Formal Languages and Applications*, chapter 13, pages 249–274. Springer, Berlin, 2004.

[29] T. Masopust. Scattered context grammars can generate the powers of 2. In *Proceedings of the 13th Conference and Competition EEICT 2007*, volume 4, pages 401–404. Faculty of Electrical Engineering and Communication, Brno University of Technology, 2007.

[30] T. Masopust and A. Meduna. On descriptional complexity of partially parallel grammars. *Fundamenta Informaticae*, 87(3):407–415, 2008.

[31] T. Masopust, A. Meduna, and J. Šimáček. Two power-decreasing derivation restrictions in generalized scattered context grammars. *Acta Cybernetica*, 18(4):783–793, 2008.

[32] T. Masopust and J. Techet. Leftmost derivations of propagating scattered context grammars: A new proof. *Discrete Mathematics and Theoretical Computer Science*, 10(2):39–46, 2008.

[33] A. Meduna. Scattered rewriting in the formal language theory. In *Missourian Annual Conference on Computing*, pages 26–36, Columbia, US, 1991.

[34] A. Meduna. Symbiotic e0l systems. artificial life: Gramatical models. *Bucharest*, pages 122–129, 1995.

[35] A. Meduna. *Automata and Languages: Theory and Applications.* Springer, London, UK, 2000.

[36] A. Meduna. Terminating left-hand sides of scattered context grammars. *Theoretical Computer Science*, 2000(237):424–427, 2000.

[37] A. Meduna. Descriptional complexity of scattered rewriting and multirewriting: An overview. *Journal of Automata, Languages and Combinatorics*, 7(4):571–577, 2002.

[38] A. Meduna. *Formal Languages and Computation: Models and Their Applications.* Taylor & Francis, New York, US, 2014.

[39] A. Meduna and J. Techet. Reduction of scattered context generators of sentences preceded by their leftmost parses. In *DCFS 2007 Proceedings*, pages 178–185, High Tatras, SK, 2007.

[40] A. Meduna and J. Techet. *Scattered Context Grammars and their Applications.* WIT Press, 2010.

[41] A. Meduna and M. Švec. *Grammars with Context Conditions and Their Applications.* Wiley, New Jersey, 2005.

[42] A. Meduna and P. Zemek. *Regulated Grammars and Their Transformations.* Faculty of Information Technology, Brno University of Technology, Brno, CZ, 2010.

[43] A. Meduna and P. Zemek. *Regulated Grammars and Automata.* Springer US, 2014.

[44] D. Milgram and A. Rosenfeld. A note on scattered context grammars. *Information Processing Letters*, 1:47–50, 1971.

[45] P. Odifreddi. *Classical Recursion Theory*, volume 2. Elsevier, 1999.

[46] E. L. Post. A variant of a recursively unsolvable problem. *Bull. Amer. Math. Soc.*, 52, 1946.

[47] G. Păun. On the index of grammars and languages. *Information and Control*, 35:259–266, 1977.

[48] G. Rozenberg and A. Salomaa, editors. *Handbook of Formal Languages, Vol. 1: Word, Language, Grammar.* Springer, New York, 1997.

[49] A. Salomaa. On the index of a context-free grammar and language. *Information and Control*, 14:474–477, 1969.

[50] A. Salomaa. *Formal Languages.* Academic Press, London, 1973.

[51] A. Salomaa. *Computation and Automata.* Cambridge Univeristy Press, Cambridge, 1985.

[52] J. Techet. A note on scattered context grammars with non-context-free components. In *3rd Doctoral Workshop on Mathematical and Engineering Methods in Computer Science*, pages 225–232. Brno University of Technology, Brno, CZ, 2007.

[53] J. Techet. *Scattered Context in Formal Languages.* PhD thesis, Faculty of Information Technology, Brno University of Technology, 2008.

[54] A. M. Turing. On computable numbers, with an application to the entscheidungsproblem. *Proceedings of the London Mathematical Society*, 42(2):230–265, 1936.

[55] G. Vaszil. On the descriptional complexity of some rewriting mechanisms regulated by context conditions. *Theoretical Computer Science*, 330(2):361–373, 2005.

[56] V. Virkkunen. On scattered context grammars. *Acta Universitatis Ouluensis*, 20(6):75–82, 1973.

[57] D. Wood. *Theory of Computation.* Wiley, New York, 1987.

Appendix A

Publications

2017

1. A. Meduna and O. Soukup. Jumping Scattered Context Grammars. Fundamenta Informaticae. Amsterdam: IOS Press, 2017, vol. 2017(152), 1-36. ISSN 0169-2968.

2016

2. A. Meduna and O. Soukup. Simple Matrix Grammars and Their Leftmost Variants. International Journal of Foundations of Computer Science. 2016, vol. 27(3), 359-373. ISSN 0129-0541.

2015

3. J. Kučera and A. Meduna and O. Soukup. Absolutely Unlimited Deep Pushdown Automata. In: Proceedings of the 10th Doctoral Workshop on Mathematical and Engineering Methods in Computer Science (MEMICS 2015). Telč: Litera, 2015, 36-44. ISBN 978-80-214-5254-1.
4. Meduna and O. Soukup. and P. Zemek. Ordered Pure Multi-Pushdown Automata. Theoretical and Applied Informatics. Warsaw: 2015, vol. 27(1), 25-47. ISSN 1896-5334.

2014

5. A. Meduna and O. Soukup. Computational Completeness Resulting from Scattered Context Grammars Working Under Various Derivation Modes. In: *Proceedings of MEMICS'14*. Brno: NOVPRESS s.r.o., 2014, 89-100. ISBN 978-80-214-5022-6.
6. O. Soukup. Leftmost Simple Matrix Grammars. In: *Sudent EEICT - Proceedings of the 20th Conference*. Brno: Brno University of Technology, 2014, 269-273. ISBN 978-80-214-4924-4.