VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

GRAMMARS WITH RESTRICTED DERIVATION TREES

PHD THESIS
DISERTAČNÍ PRÁCE

AUTHOR Ing. JIŘÍ KOUTNÝ
AUTOR PRÁCE

BRNO 2012

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

GRAMMARS WITH RESTRICTED DERIVATION TREES

GRAMATIKY S OMEZENÝMI DERIVAČNÍMI STROMY

PHD THESIS
DISERTAČNÍ PRÁCE

AUTHOR
AUTOR PRÁCE

Ing. JIŘÍ KOUTNÝ

SUPERVISOR
VEDOUCÍ PRÁCE

prof. RNDr. ALEXANDER MEDUNA, CSc.

BRNO 2012

Abstract

This doctoral thesis studies theoretical properties of grammars with restricted derivation trees. After presenting the state of the art concerning this investigation area, the research is focused on the three main kinds of the restrictions placed upon the derivation trees. First, it introduces completely new investigation area represented by cut-based restriction and examines the generative power of the grammars restricted in this way. Second, it investigates several new properties of path-based restriction placed upon the derivation trees. Specifically, it studies the impact of *erasing productions* on the generative power of grammars with restricted path and introduces two corresponding normal forms. Then, it describes a new relation between grammars with restricted path and some pseudoknots. Next, it presents a counterargument to the generative power of grammars with controlled path that has been considered as well-known so far. Finally, it introduces a generalization of path-based restriction to not just one but several paths. The model generalized in this way is studied, namely its pumping, closure, and parsing properties.

Abstrakt

V této disertační práci jsou studovány teoretické vlastnosti gramatik s omezenými derivačními stromy. Po uvedení současného stavu poznání v této oblasti je výzkum zaměřen na tři základní typy omezení derivačních stromů. Nejprve je představeno zcela nové téma, které je založeno na omezení řezů a je zkoumána vyjadřovací síla takto omezené gramatiky. Poté je zkoumáno několik nových vlastností omezení kladeného na cestu derivačních stromů. Zejména je studován vliv vymazávacích pravidel na vyjadřovací sílu gramatik s omezenou cestou a pro tyto gramatiky jsou zavedeny dvě normální formy. Následně je popsána nová souvislost mezi gramatikami s omezenou cestou a některými pseudouzly. Dále je prezentován protiargument k vyjadřovací síle tohoto modelu, která byla dosud považována za dobře známou vlastnost. Nakonec je zavedeno zobecnění modelu s omezenou cestou na ne jednu, ale několik cest. Tento model je následně studován zejména z hlediska vlastností vkládání, uzávěrových vlastností a vlastností syntaktické analýzy.

Keywords

tree controlled grammars, level controlled grammars, path controlled grammars, paths controlled grammars, cut controlled grammars, ordered cut controlled grammars, regulated rewriting, restricted derivation trees

Klíčová slova

stromem řízené gramatiky, úrovněmi řízené gramatiky, cestou řízené gramatiky, cestami řízené gramatiky, řezy řízené gramatiky, uspořádanými řezy řízené gramatiky, řízené přepisování, omezené derivační stromy

Citation

Jiří Koutný: Grammars with Restricted Derivation Trees, disertační práce, Brno, Faculty of Information Technology, Brno University of Technology, 2012

Citace

Jiří Koutný: Gramatiky s omezenými derivačními stromy, phd thesis, Brno, FIT VUT v Brně, 2012

Grammars with Restricted Derivation Trees

Declaration

I hereby declare that this doctoral thesis is my own work that has been created under the supervision of prof. RNDr. Alexander Meduna, CSc. In order to create a compact and systematic study, I have included and duly acknowledged the results established by some other people. Specifically, Part II being a brief summary of the state of the art in the restricted-derivation-tree-related investigation area. Some results were established in cooperation with my colleagues. More precisely, the results of Sec. 9.3.2 and Sec. 9.3.3 are a joint work with Ing. Zbyněk Křivka, Ph.D and the results of Sec. 9.3.4 are a joint work with Ing. Martin Čermák. All of the other results presented through Part III are my own and they were established in collaboration only with my supervisor.

....
Ing. Jiří Koutný
June 12, 2012

Acknowledgements

I am indebted to many people for their assistance in various aspects of creating this thesis. My paramount thanks go to Alexander Meduna, the supervisor of my doctoral as well as master studies, for his pedagogical and scientific leadership and support during the whole research. I have greatly benefited from the conversation with many of my colleagues at Brno University of Technology. I would like to thank especially Zbyněk Křivka for his endless discussions and wonderful comments. My special thanks belong to Masami Ito, Shinnosuke Seki, and Szilard Zsolt Fazekas for their inspirational talks and invaluable comments as well as to Erzsébet Csuhaj-Varjú and Gyuri Vaszil who tried to give me at least a small part of their knowledge. I gladly acknowledge the support of all grants I have participated on. Specifically, the CZ.1.05/1.1.00/02.0070, CZ.1.07/2.4.00/12.0017, FIT-S-10-2, FIT-S-11-2, FR2581/2010/G1, GD102/09/H042, MEB041003, and MSM0021630528 grants. Most importantly, I thank my family for its constant support during all of my studies.

Contents

Chapter 1

Introduction

The *formal language theory* is an inherent part of the *theoretical computer science* particularly concerned with the study of the formal models. The formal models are mathematical objects used to describe the formal languages. The fundamental models include grammars and automata. The former are used to generate words and the latter accept them.

Grammars are the kind of the *rewriting models* that start from a specified symbol (i.e., start symbol). Then, the symbol is modified according to the given set of rewriting productions. Each production is composed of two components—the left-hand side and the right-hand side of a production. The application of a production on a word is done by rewriting a symbol equivalent to the left-hand side of a production by its right-hand side in the word. This process is known as a *derivation step*. During the computation of a derivation step, just one symbol is rewritten in the word. Given a start symbol of a grammar, a derivation step is computed repeatedly by applying the productions from the given set. Once the resulting word is composed of the symbols that cannot be rewritten anymore, the process of applying derivation steps ends and the resulting word belongs to the language of the grammar.

Essentially, the grammars are composed of a finitely many symbols that are rewritten by finitely many production in finitely many derivation steps. In this way, the grammars represent a finite description of even infinite languages. By the notion of infinite languages are meant those languages that contains infinitely many words. Since the most of the languages commonly used in practice are infinite, the grammars represent a powerful tool how to deal with them. In the formal language theory, there exists a huge variety of grammars which essentially differ in two domains. Specifically, in the complexity of the productions and in the way how to select appropriate production to be applied in a derivation step.

Generally, the complexity of rewriting productions can be seen from two angles—theoretical and practical.

- Theoretical viewpoint: As little as possible restrictions placed on the form of the rewriting productions in a rewriting model is desirable. More specifically, the more complex rewriting productions are, the larger class of languages may the model generate. In other words, to generate complex languages, complex productions are needed. By the notion of a form of a production, namely the number of the symbols on its left-hand side is meant.

- Practical viewpoint: Grammars are theoretical models that are implemented in many practical applications. From the perspective of cost-effective implementation, the simple rewriting productions are desirable. As simple-enough productions for effective implementation, those of the form with just one symbol on their left-hand side are considered. Such kind of productions are referred to as *context-free productions*, since they can be applied without any consideration of a context of currently rewritten symbol.

Non-regulated rewriting models like grammars and automata belong to the well-known core of the formal language theory and they are frequently used in practice. Indeed, automata including its variants underlie lexical analysers (see [4] and [90]), context-free grammars represent the basis of both top-down as well as bottom-up parsers (see [4] and [5]), etc. However, the power of the models with simple productions is indeed smaller than required for usage in many practical applications. On the other hand, the models that use only simple productions are usually easier to implement. As a background, it is desirable to extend context-free grammars as in many applications there are some natural phenomena which cannot be captured by context-free rewriting. More precisely, the motivation is based on the observation that many of the languages commonly used in practice, including programming and natural languages, are not context-free (see [49], [56], [107], [108], and [116]). Consequently, that means such languages cannot be generated by a grammar with only context-free productions. For these reasons, the idea whether or not it would be possible to use grammars with only context-free productions and increase the corresponding power in some other way—without changing the form of rewriting productions. Basically, this can be achieved by two fundamental approaches—using a kind of a regulation of rewriting or using more than one grammar with context-free productions in a model:

- Using a kind of a regulation of rewriting. By the notion of a *regulation*, the way how to select appropriate production to be applied is meant. Indeed, a situation is common in which, given a word, it is possible to apply several productions. Informally, the essential idea is represented by the observation that a regulation mechanism somehow prescribes the order of productions the grammar must follow. Therefore, many different kinds of such a regulation have been introduced in order to ensure selecting appropriate production. All of the resulting models based on a kinds of regulation are collectively referred to as *regulated rewriting models.*

- Using more than one grammar with context-free productions in a model. Roughly speaking, the main underlying idea is based on the observation that from the cooperation of several simple models, we can obtain more power than from each of them if they work separately. These systems were also thoroughly studied and the corresponding investigation area is referred to as the *theory of grammar systems.* However, we will deal with them only rarely in this work.

Informally, the goal of this work is to introduce a model that generates more than context-free languages and is usable in practice. From the theoretical viewpoint it means, the model should be able to generate namely the programming languages and the languages used in linguistics (e.g., multiple repetition, cross-dependencies, and copy-language). From the practical viewpoint, it should be possible to develop sophisticated parsing methods working in a polynomial time for the model.

One way to extend the power of context-free grammars is to consider *context-sensitive grammars* where the productions are more complex. Indeed, context-sensitive grammars contain the productions with even more than a single symbol on the left-hand side. However, despite their great power, generating complex languages by context-sensitive grammars actually leads to several fundamental problems making their practical usage problematic (see [12], [22], [89], [109], [130], and [134]). Specifically, for context-sensitive grammars, many problems are undecidable, it is difficult to describe the derivation by a graph structure, etc.

One of the many others approaches extending the power of context-free grammars is represented by *matrix grammars* introduced by Abraham in [1]. The fundamental underlying principle in a derivation step in matrix grammars is that not just one but a fixed number of context-free productions are required to be applied in a given order. This provides synchronization among different parts of a generated word and many non-context-free languages can be generated in this way (see [101], [114], and[139]).

There are lots of other well-known approaches for extending the power of context-free grammars which preserve the context-free nature of productions. Specifically, *Random Context Grammars* (see [125]), *Programmed Grammars* (see [111]), *Ordered Grammars* (see [41]), *Indian and Russian Parallel Grammars* (see [73]), *Indexed Grammar* (see [3]), and many others. However, these approaches do not represent the main topic of this work although some connections can probably be found.

1.1 Derivation Tree Restricted Models

One of the power-extending approaches is represented by the restrictions placed upon the derivation trees. Given a grammar, by the notion of a *derivation tree*, a graph structure depicting the application of productions on the start symbol up to the resulting word is meant. Indisputably, the investigation of context-free grammars with restricted derivation trees represents an important trend in today's formal language theory (see [17], [24], [27], [29], [70], [63], [65], [66], [68], [80], [81], [82], [97], [104], [122], and [123]). In essence, these grammars generate their languages just as ordinary context-free grammars do (see Def. 3.35) but their derivation trees (see Def. 3.44) must satisfy some simple prescribed conditions.

The following two sections give an informal overview of the results related to the investigation of derivation-tree-restricted grammars. Based on this informal description, Part II being a strictly formal summary of the results presented here. The definitions needed just to present the results of the other authors are omitted in this work. However, the appropriate references for the definitions are given. Through this section, we assume the reader is familiar with the fundamentals of the formal language theory (see Part I). Several results concerning the derivation-tree-restricted models related to L-systems (see [60], [61], [62], [76], [74], [75], and [106]) are not included in this work since the investigation presented in Part III focuses rather on the sequential (i.e., grammars) than parallel rewriting (i.e., L-systems).

1.1.1 Level Based Restriction

The idea of restrictions placed upon the derivation trees of context-free grammars is introduced by Culik and Maurer in [24] and the resulting grammars restricted in this way are referred to as *tree controlled grammars* (see Def. 4.1). In essence, the notion of a tree controlled grammar is defined as follows: take a context-free grammar, G, and a regular language, R. A word, w, generated by G belongs to the language defined by G and R if there is a derivation tree, t, for w in G such that all levels (see Def. 2.18) of t (except the last one) are described by R. Given a tree controlled grammar, (G, R), G and R are referred to as controlled grammar and control language, respectively. Culik and Maurer investigate several basic properties of tree controlled grammars—namely, the membership problem (see Th. 4.1) and the generative power (see Th. 4.2, Th. 4.3, Th. 4.4, Col. 4.5, and Th. 4.6).

Based on the original definition of a tree controlled grammar, Păun studies the modifications where many well-known types of both controlled grammars and control languages are considered in [104]. More precisely, Păun studies controlling the levels of the derivation trees of a regular grammar by several types of a control language (see Th. 4.7 and Col. 4.8), controlling the levels of the derivation trees of a context-free grammar without erasing productions by several types of a control language (see Th. 4.9, Th. 4.10, Col. 4.11, Th. 4.12, and Col. 4.13), and controlling the levels of the derivation trees of a context-free grammar by a finite language (see Th. 4.14 and Col. 4.15).

It is well-known that tree controlled grammars with a context-free grammar controlled by a regular language characterize the class of recursively enumerable languages (see Th. 4.5). Thus, the question arises whether or not it is possible to achieve the same generative power as tree controlled grammars have when the levels of the derivation trees are restricted by a subregular control language (see Sec. 3.7). This problem is studied by Dassow and Truthe in [29], where many types of subregular languages are considered. Dassow and Truthe study primarily controlling the levels of the derivation trees of a context-free grammar by two types of a language such that one is a subset of the other (see Lem. 4.16) and controlling the levels of the derivation trees of a context-free grammar by many different types of subregular languages (see Th. 4.17, Th. 4.18, Th. 4.19, Col. 4.20, Th. 4.21, Th. 4.22, Th. 4.23, and Th. 4.24). The same authors, Dassow and Truthe, also study hierarchies of subregularly tree controlled languages in [27] and [28]. They present controlling the levels of the derivation trees of a context-free grammar by the union of monoids (see Th. 4.25), by regular languages with restricted descriptional complexity (see Lem. 4.26, Th. 4.27, and Th. 4.28), and by the language accepted by a deterministic finite automaton with at most given number of states (see Th. 4.29).

Stiebe in [118] states that there is a tree controlled grammar for every linearly bounded queue automaton (see Lem. 4.30). Then, Stiebe proves that controlling the levels of the derivation trees of a context-free grammar by the language accepted by a minimal finite automaton with at most five states characterize the class of context-sensitive languages (see Th. 4.31). If, additionally, erasing productions in a controlled grammar are allowed, controlling the levels of the derivation trees of a context-free grammar by the language accepted by a minimal finite automaton with at most five states characterizes the class of recursively enumerable languages (see Th. 4.32).

Turaev, Dassow, and Selamat in [122] examine tree controlled grammars with bounded nonterminal complexity and demonstrate that nine/seven nonterminals in a tree controlled grammar are enough to generate any recursively enumerable language (see Th. 4.33 and Th. 4.34). Then, they establish that three nonterminals in a tree controlled grammar are enough to generate any regular language (see Th. 4.35) and any regular simple matrix language can be generated by a tree controlled grammar (see Th. 4.36) with three nonterminals. Finally, they demonstrate that three nonterminals in a tree controlled grammar are enough to generate any linear language (see Th. 4.37). The same authors in [123] state several further nonterminal complexity related properties of tree controlled grammars (see Lem. 4.38).

A strictly formal summary of the results concerning tree controlled grammars can be found in Chap. 4.

1.1.2 Path Based Restriction

As an attempt to increase the power of context-free grammars without changing the basic formalism and loosing some basic properties of context-free languages (decidability, efficient parsing, etc.), Marcus, Martín-Vide, Mitrana, and Păun in [80] study a new type of a restriction in a derivation: a derivation tree in a context-free grammar is accepted if it contains a path described by a control language. More precisely, they consider two context-free grammars, G and G'. A word, w, generated by G belongs to the language defined by G and G' if there is a derivation tree, t, for w in G such that there exists a path of t described by the language of G'. Based on this restriction, they introduce a *path controlled grammar* (see Def. 5.2) and study several properties of this model. Specifically, they study controlling a path of the derivation trees of several types of grammars by a regular language (see Th. 5.1) and controlling a path of the derivation trees of a regular grammar by a linear or context-free language (see Th. 5.3 and Th. 5.4). Then, they establish two kinds of pumping properties depending on the type of a controlled grammar (see Th. 5.6, and Th. 5.7), some consequences to the closure properties of path controlled grammars (see Th. 5.9, Th. 5.10, Th. 5.11), and a basic parsing property for path controlled grammars (see Th. 5.12). They also investigate the generative power of path controlled grammars (see Th. 5.5 and Col. 5.8). However, there exists a serious problem with the correctness of the proof they present (see our discussion in Sec. 8.3.4).

As a continuation of the investigation of path-based restrictions, Martín-Vide and Mitrana study parsing properties of path controlled grammars (see Th. 5.13), closure properties of path controlled grammars (see Th. 5.14, Th. 5.15, and Th. 5.16), and several decision problems for path controlled grammars (see 5.17) in [81] and [82].

For a strictly formal summary of the results related to path controlled grammars, see Chap. 5.

1.2 Goals of the Thesis

As it clearly follows from the previous sections, level-based restriction is well-studied and the most of the important questions have been answered. On the other hand, in the case of path-based restriction many basic properties including the generative power have not been successfully investigated yet. Moreover, several other restrictions placed upon the derivation trees have not yet been introduced at all. Indeed, the restrictions placed upon the cuts of the derivation tree (see Chap. 7) as well as upon several paths of the derivation trees (see Chap. 9) represent completely new investigation areas.

Thus, the goals of the doctoral thesis consist in three investigation areas. First, to introduce new investigation area represented by cut-based restrictions and establish the generative power of the model restricted in this way. Second, to establish several new results in the investigation of one-path-restricted grammars introduced in [80]. Third, to generalize one-path-restricted model into several paths and investigate several its properties.

Since each of these kind of restrictions represents relatively independent derivation-tree-restriction-related topic, each of them contains its own motivation (see Sec. 7.1, Sec. 8.1, and Sec. 9.1) and further research ideas sections (see Sec. 7.3.1, Sec. 8.3.1, Sec. 8.3.2, Sec. 8.3.3, Sec. 8.3.4, Sec. 9.3.1, Sec. 9.3.2, Sec. 9.3.3, and Sec. 9.3.4).

1.3 Chapter Survey

This doctoral thesis consists of 3 parts that are subsequently divided into 10 chapters.

- Part I, *Notation and Basic Definitions*, contains two chapters—*Basic Mathematical Definitions* and *Language and Rewriting*. The first being the summary of essential well-known definitions related to the *set theory* and the *graph theory*. The second being the overview of the fundamental definitions and models of the *formal language theory* and the *theoretical computer science* needed in the subsequent parts of this work. For the conciseness, in several special cases where a complete list of definitions should be given just to formally define one specific property, we present only the reference for those definitions and define the required property directly. However, these cases are almost rare and they are always duly pointed out.

- Part II, *State of the Art*, being the formal summary of the crucial derivation-tree-based restriction results. All results are presented using the terminology introduced in Part I. Except of one that is controversial (see Sec. 8.3.4), all proofs related to the presented results are omitted. However, appropriate reference where one can found the proof if required is always included.

- Part III being the main part of this doctoral thesis and it is composed of five chapters.
  - The first reformulates the fundamental definitions so that all derivation-tree-based restrictions can be studied using the same notation. Then, this chapter summarizes the preliminaries common for all new results presented in the subsequent chapters of this work.
  - The second, *Cut Tree Controlled Grammars*, introduces a new derivation-tree-based restriction, the rewriting model based upon aforementioned restriction, and several properties related namely to its generative power.
  - The next chapter, *Path Tree Controlled Grammars*, deals with the one-path-restricted rewriting model as it is presented informally in Sec. 1.1.2 and formally in Chap. 8. It introduces new results related to the *normal forms* and the presence of erasing productions in a controlled grammar. Then, this chapter presents a relationship between biology and the formal language theory in the form of word representation of pseudoknots generated by path controlled grammars. Last section of this part being a counterargument against the proof of the generative power of path controlled grammars that has been considered as correct so far.
  - The chapter *n-Path Tree Controlled Grammars* being a generalization of path-restricted rewriting model to a restriction placed upon not just one but several paths. Then, it presents several properties of the model restricted in this way. Specifically, the generative power of a kind of all-paths-restricted rewriting model, closure and pumping properties in relation to the number of controlled paths, and the approximation of the generative power for n-path restricted model. The last section of this part presents an application related result concerning the parsing of path restricted languages.
  - Last chapter, *Summary*, being a brief resume of the most important achieved results deeply studied in the aforementioned sections and written or co-written by the author of this doctoral thesis.

Part I

Notation and Basic Definitions

Chapter 2

Basic Mathematical Definitions

In this two-section chapter, we introduce all fundamental definitions needed for presenting the current state of the art as well as for the new results in the investigation of grammars with restricted derivation tress. Since all definitions presented in this chapter belong to the well-known core of mathematical knowledge, they are presented formally without any comments. However, for those who need more detailed explanation including several examples, the bibliographical remarks (see Sec. 3.10) are included. For better readability, several conventions are introduced.

2.1 Set Theory

The formal language theory is based namely upon the *set theory.* Indeed, formal languages are sets of words. This section summarizes the definitions of the notions related to the set theory.

Definition 2.1 (Set)**.** A *set*, Σ, is a collection of the elements, which are taken from some prespecified universe. If a set, Σ, contains an element, a, then this is denoted as $a \in \Sigma$ and refers to a as a member of Σ. However, if a does not belong to Σ, this is denoted as $a \notin \Sigma$.

Definition 2.2 (Cardinality of a set)**.** Let Σ be a set. The *cardinality* of Σ, $\mathrm{card}(\Sigma)$, is the number of members of Σ. The set that has no member is the *empty set*, denoted $\emptyset$, and $\mathrm{card}(\emptyset) = 0$.

Definition 2.3 (Finite and infinite set)**.** If a set, Σ, has a finite number of members, then Σ is a *finite set*; otherwise, Σ is an *infinite set*. A finite set Σ is customarily specified by listing its members; that is, $\Sigma = \{a_1, a_2, \dots, a_n\}$ where all $a_i \in \Sigma$, for $1 \leq i \leq n$, are all members of Σ. An infinite set, Ω, is usually specified by a property, π, so that Ω contains all elements satisfying π; this specification has the following general format: $\Omega = \{a : \pi(a)\}$.

Definition 2.4 (Subset and proper subset)**.** Let Σ and Ω be two sets. Σ is a *subset* of Ω, denoted as $\Sigma \subseteq \Omega$, if each member of Σ belongs to Ω. Σ is a *proper subset* of Ω, denoted as $\Sigma \subset \Omega$, if $\Sigma \subseteq \Omega$ and Ω contains an element, a, such that $a \notin \Sigma$. Σ *equals* Ω, denoted as $\Sigma = \Omega$, if $\Sigma \subseteq \Omega$ and $\Omega \subseteq \Sigma$.

Definition 2.5 (Difference of sets)**.** Let Σ_1 and Σ_2 be two sets. The *difference* of Σ_1 and Σ_2, symbolically written as $\Sigma_1 - \Sigma_2$, is defined as $\Sigma_1 - \Sigma_2 = \{x : x \in \Sigma_1 \text{ and } x \notin \Sigma_2\}$.

Relations

Definition 2.6 (Ordered pair)**.** Let a and b be two elements. Then, (a, b) denotes the *ordered pair* consisting of a and b in this order. Let (a, b) and (c, d) be two ordered pairs. Then, $(a, b) = (c, d)$ if and only if $a = c$ and $b = d$.

Definition 2.7 (Cartesian product)**.** Let Σ and Ω be two sets. The *Cartesian product* of Σ and Ω, denoted as $\Sigma \times \Omega$, is defined as $\Sigma \times \Omega = \{(a, b) : \ a \in \Sigma \text{ and } b \in \Omega\}$.

Definition 2.8 (Binary relation)**.** Let Σ and Ω be two sets. A *binary relation* ρ from Σ to Ω is a subset of $\Sigma \times \Omega$; that is, $\rho \subseteq \Sigma \times \Omega$. If $\Sigma = \Omega$, then ρ is a *relation on* Σ.

Conventions

Instead of $(a, b) \in \rho$, we usually write $a \in \rho(b)$ or $a\rho b$.

Definition 2.9 (Partial order)**.** Let P be a set. A *partial order* on a nonempty set P is a binary relation $\leq$ on P such that for all $x, y, z \in P$: $x \leq x$ (reflexive), $x \leq y, y \leq x$ imply $x = y$ (antisymmetric), and $x \leq y \leq z$ imply $x \leq z$ (transitive). The pair $(P, \leq)$ is called a *partially ordered set.*

Definition 2.10 (Total order)**.** Let P be a set and $\leq$ be a binary relation. A partially ordered set $(P, \leq)$ is *totally ordered* if for every $x, y \in P$: $x \leq y$ or $y \leq x$.

Definition 2.11 (k-fold product)**.** Let ρ be a binary relation on a set, Σ, and let k be a natural number. The *k-fold product* of a binary relation, denoted as ρ^k, is defined as $a\rho^1 b$ if and only if $a\rho b$; for $k \geq 2$, $a\rho^k b$ if and only if there exists $c \in \Sigma$ such that $a\rho c$ and $c\rho^{k-1}b$.

Definition 2.12 (Transitive closure)**.** Let ρ be a binary relation on a set, Σ. The *transitive closure* of ρ, denoted as ρ^+, is defined as follows: $a\rho^+ b$ if and only if $a\rho^i b$ for some $i \geq 1$. Consequently, $a\rho^+ b$ if and only if for some $n \geq 0$, $a\rho c_1, c_1\rho c_2, \ldots, c_{n-1}\rho c_n, c_n\rho b$ where $c_1, c_2, \ldots, c_n \in \Sigma$ (the case when $n = 0$ actually means $a\rho b$).

Definition 2.13 (Reflexive and transitive closure)**.** Let ρ be a binary relation on a set, Σ. The *reflexive and transitive closure* of ρ, denoted as ρ^*, is defined as $a\rho^*$ if and only if $a\rho^+ b$ or $a = b$.

Definition 2.14 (Function)**.** Let Σ and Ω be two sets, and let ϕ be a binary relation from Σ to Ω such that for every $a \in \Sigma$, $\text{card}(\{b : \ b \in \Omega \text{ and } a\phi b\}) \leq 1$. Then ϕ is a *function* from Σ to Ω.

2.2 Graph Theory

The process of generating the formal languages is usually captured by the graphs. Therefore, in this section, we summarize basic notions of the *graph theory.*

Definition 2.15 (Graph)**.** Let Σ be a set. A *graph* is a pair, $G = (\Sigma, \rho)$, where ρ is a binary relation on Σ. Consider a graph, $G = (\Sigma, \rho)$. Members of Σ are called *nodes*, and ordered pairs in ρ are called *edges*. If $(a, b) \in \rho$, then (a, b) *leaves* a and *enters* b. Let $a \in \Sigma$. Then, *in-degree* of a equals $\text{card}(\{b : \ (b, a) \in \rho\})$ and *out-degree* of a equals $\text{card}\{c : \ (a, c) \in \rho\})$. A sequence of nodes, $a_0a_1 \ldots a_n$, where $n \geq 1$, is a a *graph-path of length* n from a_0 to a_n if $(a_{i-1}, a_i) \in \rho$ for all $i = 1, \ldots, n$; if, in addition, $a_0 = a_n$ then $a_0a_1 \ldots a_n$ is a *cycle of length* n.

Conventions

For brevity, a *graph* is called a *directed graph*.

Definition 2.16 (Acyclic graph)**.** An *acyclic graph* is a graph, $G = (\Sigma, \rho)$, that has no cycles. If $a \in \Sigma$ is a node having out-degree 0, then a is a *leaf*. If $(a_0, a_1, \dots, a_n)$ is a graph-path in G, then a_n is a *descendent* of a_0; in addition, if $n = 1$, then a_n is a *direct descendent* of a_0.

Trees

Definition 2.17 (Tree)**.** A *tree* is an acyclic graph, $G = (\Sigma, \rho)$, satisfying these three properties: G has a specified node whose in-degree is 0; this node represents the *root* of G, denoted by $\mathrm{root}(G)$; if $a \in \Sigma$ and $a \neq \mathrm{root}(G)$, then a is a descendent of $\mathrm{root}(G)$ and the in-degree of a is 1; and each node $a \in \Sigma$ which is not a leaf has its direct descendent, b_1 through b_n, ordered from the left to the right so that b_1 is the leftmost direct descendent of a and b_n is the rightmost direct descendent of a.

Definition 2.18 (Level, path, cut, frontier, depth, elementrary tree, and subtree)**.** Let $G = (\Sigma, \rho)$ be a tree.

A *level*, l, of G is a sequence, s, of all nodes with the same distance from $\mathrm{root}(G)$. In other words, a level, l, is a sequence, $s = n_1 n_2 \dots n_k$, such that there is a graph-path of length ℓ in G for all sequences $\mathrm{root}(G) \dots n_i$, for $1 \leq i \leq k$ and some $l \geq 1$.

A *path*, p, of G is a sequence, s, of nodes where the first node of s is $\mathrm{root}(G)$, last node of s is a leaf, and there is an edge in G between each two consecutive nodes of s. By other words, a path, p, is a sequence, $s = n_1 n_2 \dots n_k$, such that s is a graph-path of length k in G with $n_1 = \mathrm{root}(G)$, and n_k is a leaf of G, for some $k \geq 1$.

A *cut*, c, of G is a sequence, s, of the nodes such that each path of G has precisely one node in c. By other words, a cut, c, is a sequence, $s = n_1 n_2 \dots n_k$, such that for each path, $p = m_1 m_2 \dots m_\ell$, of G, $\mathrm{card}\{\{n_1, n_2, \dots, n_k\} \cap \{m_1, m_2, \dots, m_\ell\}\} = 1$, for $k, \ell \geq 1$.

The *frontier* of G, $\mathrm{fr}(G)$, is the sequence of G's leaves ordered from left to right.

The *depth* of G, $\mathrm{depth}(G)$, is the length of the longest path in G; if $\mathrm{depth}(G) = 1$, then G is an *elementary tree*.

If $G' = (\Sigma', \rho')$ represents a tree satisfying these four conditions: $\Sigma' \neq \emptyset$; $\Sigma' \subseteq \Sigma$; $\rho' = (\Sigma' \times \Sigma') \cap \rho$; and in G, no node in $\Sigma - \Sigma'$ is a descendant of a node in Σ', then G' is a *subtree* of G.

Definition 2.19 (Word representation of sequence of nodes)**.** For a sequence, s, of the nodes of a derivation tree, the *word obtained by concatenating all symbols of* s is denoted as $\mathrm{word}(s)$.

Conventions

Let $G = (\Sigma, \rho)$ be a tree. G is usually described pictorially with each node, $a \in \Sigma$, represented by a circle, and each edge, $(a, b) \in \rho$, represented by an arrow from the circle a to the circle b. For simplicity, we draw a tree, G, with its root on the top and all edges directed down—thus, the arrows are omitted.

Chapter 3

Languages and Rewriting

Based upon the definitions of the previous chapter, we formulate the basics of the *formal language theory*. After that, we present the definitions of the language-describing models needed in the following parts of this work.

3.1 Formalization of Languages

Formal languages are defined as sets of words. Therefore, except of the basic set-related notions presented in the previous chapter, we introduce several language-related definitions here.

Alphabets and Words

Definition 3.1 (Alphabet and symbol)**.** An *alphabet* is a finite, nonempty set of elements, which are called *symbols*.

Definition 3.2 (Word)**.** Let Σ be an alphabet, then ε is a word over Σ; if x is a word over Σ and $a \in \Sigma$, then xa is a word over Σ. Let Σ^* denote the set of all words over Σ and set $\Sigma^+ = \Sigma^* - \{\varepsilon\}$.

Definition 3.3 (Length of word)**.** Let x be a word over an alphabet, Σ. The length of x, $|x|$, is defined as if $x = \varepsilon$, then $|x| = 0$; if $x = a_1a_2 \dots a_n$, for some $n \geq 1$, where $a_i \in \Sigma$ for all $1 \leq i \leq n$, then $|x| = n$.

Definition 3.4 (Concatenation of words)**.** Let x and y be two words over an alphabet, Σ. Then, xy is the *concatenation* of x and y. For every word, x, it holds $x\varepsilon = \varepsilon x = x$.

Definition 3.5 (Reversal of word)**.** Let x be a word over an alphabet, Σ. The *reversal* of x, reversal(x), is defined as if $x = \varepsilon$, then reversal$(x) = \varepsilon$; if $x = a_1a_2 \dots a_n$, for some $n \geq 1$, and $a_i \in \Sigma$ for $1 \leq i \leq n$, then reversal$(a_1a_2 \dots a_n) = a_n \dots a_2a_1$.

Definition 3.6 (Subword)**.** Let x and y be two words over an alphabet, Σ. Then, x is a *subword* of y if there exist two words, z and z', over Σ so $zxz' = y$; moreover, if $x \notin \{\varepsilon, y\}$ then x is a *proper subword* of y.

Languages

Definition 3.7 (Language)**.** Let Σ be an alphabet and let $L \subseteq \Sigma^*$. Then, L is a *language* over Σ.

Definition 3.8 (Minimal alphabet of word)**.** Let Σ be an alphabet. For a word, $w \in \Sigma^*$, $\mathrm{alph}(w)$ denotes the set of all symbols of Σ occurring in w.

Definition 3.9 (Minimal alphabet of language)**.** Let Σ be an alphabet and $L \subseteq \Sigma^*$. Then, $\mathrm{alph}(L) = \bigcup_{w \in L} \mathrm{alph}(w)$ denotes *minimal alphabet of* L.

Definition 3.10 (Finite and infinite language)**.** Let L be a language. L is *finite* if $\mathrm{card}(L) = n$ for some $n \geq 0$; otherwise, L is *infinite*.

Definition 3.11 (Class of finite languages)**.** Class of *finite languages* is defined as $\mathbf{FIN} = \{L : L \text{ is a finite language}\}$.

Definition 3.12 (Concatenation of languages)**.** Let L_1 and L_2 be two languages. The *concatenation* of L_1 and L_2, L_1L_2, is defined as $L_1L_2 = \{xy : x \in L_1 \text{ and } y \in L_2\}$.

Definition 3.13 (Union of languages)**.** Let L_1 and L_2 be two languages. The *union* of L_1 and L_2, $L_1 \cup L_2$, is defined as $L_1 \cup L_2 = \{x : x \in L_1 \text{ or } x \in L_2\}$.

Definition 3.14 (Intersection of languages)**.** Let L_1 and L_2 be two languages. The *intersection* of L_1 and L_2, $L_1 \cap L_2$, is defined as $L_1 \cap L_2 = \{x : x \in L_1 \text{ and } x \in L_2\}$.

Definition 3.15 (Difference of languages)**.** Let L_1 and L_2 be two languages. The *difference* of L_1 and L_2, $L_1 - L_2$, is defined as $L_1 - L_2 = \{x : x \in L_1 \text{ and } x \notin L_2\}$.

Definition 3.16 (Complement of language)**.** Let L be a language over an alphabet Σ. The *complement* of L, $\overline{L}$, is defined as $\overline{L} = \Sigma^* - L$.

Definition 3.17 (Power of language)**.** Let L be a language. For $i \geq 0$, the ith *power* of L, L^i, is defined as $L^0 = \varepsilon$; for all $i \geq 1$, $L^i = LL^{i-1}$.

Definition 3.18 (Closure of language)**.** Let L be a language. The *closure* of L, L^*, is defined as $L^* = \bigcup_{i=0}^{\infty} L^i$.

Translations

Definition 3.19 (Translation)**.** Let Σ be an *input alphabet*, and let Ω be an *output alphabet*. A *translation*, τ, from Σ^* to Ω^* is a binary relation from Σ^* to Ω^*.

Definition 3.20 (Substitution)**.** Let Σ and Ω be two alphabets, and let τ be a translation from Σ^* to Ω^* such that for all $x, y \in \Sigma^*$, $\tau(xy) = \tau(x)\tau(y)$. Then, τ is a *substitution*.

Definition 3.21 (Homomorphism)**.** Let Σ and Ω be two alphabets, and let τ be a substitution from Σ^* to Ω^*. If τ represents a function from Σ^* to Ω^*, then τ is a *homomorphism*.

3.2 Unrestricted Languages

As the most powerful language-generating model, we present the notion of *unrestricted grammars* and several related definitions. The great generative power of the aforementioned model is achieved by unrestricted form of the rewriting productions.

Definition 3.22 (Unrestricted grammar)**.** *Unrestricted grammar* is a quadruple $G = (V, T, P, S)$ where V is a total alphabet, T is an alphabet of terminals such that $T \subset V$, $P \subseteq V^*(V - T)V^* \times V^*$ is a finite binary relation, $S \in V - T$ is the start symbol.

Conventions

For the conciseness, see Conventions in Sec. 3.35 for context-free grammars and apply them for unrestricted grammars analogously.

Definition 3.23 (Derivation)**.** Let $G = (V, T, P, S)$ be an unrestricted grammar, $p \in P$, and $x, y \in V^*$. Then, $x\,\text{lhs}(p)y$ *directly derives* $x\,\text{rhs}(p)y$ according to p in G; symbolically, $x\,\text{lhs}(p)y \Rightarrow x\,\text{rhs}(p)y\,[p]$ or, briefly, $x\,\text{lhs}(p)y \Rightarrow x\,\text{rhs}(p)y$. By analogy with the corresponding definition for a context-free grammar (see Def. 3.37), for all $n \geq 0$, define $v \Rightarrow^n w\,[\pi]$ in G, $v \Rightarrow^+ w\,[\pi]$, and $v \Rightarrow^* w\,[\pi]$.

Definition 3.24 (Generated language)**.** Let $G = (V, T, P, S)$ be an unrestricted grammar. The *language generated* by G, $L(G)$, is defined as $L(G) = \{w : \ w \in T^* \text{ and } S \Rightarrow^* w \text{ in } G\}$.

Definition 3.25 (Recursively enumerable language)**.** A language, L, is *recursively enumerable* if there is an unrestricted grammar, G, such that $L(G) = L$.

Definition 3.26 (Class of recursively enumerable languages)**.** The *class of recursively enumerable languages* is defined as $\mathbf{RE} = \{L : \ L = L(G) \text{ for an unrestricted grammar } G\}$.

Definition 3.27 (Pentonnen normal form for unrestricted grammars)**.** An unrestricted grammar, $G = (V, T, P, S)$, is in *Pentonnen normal form* if every production, $p \in P$, has one of these four forms: $AB \to AC$, $A \to BC$, $A \to a$, or $A \to \varepsilon$ with $A, B, C \in V - T$ and $a \in T$.

Definition 3.28 (Recursive language)**.** A language L is *recursive* if and only if both L and $\overline{L}$ are recursively enumerable languages.

Definition 3.29 (Decidable and undecidable problems)**.** Given a yes/no question Q, a language, L, can be built by taking all the instances of Q where the answer is yes (with Q converted to a word somehow). A yes/no question is *decidable* if the associated language is recursive. Let G be a grammar. Then, the *non-emptiness problem*, *finiteness problem*, and *membership problem* is represented by the question „Is $L(G) \neq \emptyset$?“, „Is $L(G)$ finite?“, and „Is $w \in L(G)$?“, respectively.

Definition 3.30 (Time complexity)**.** Let M be a Turing machine (see Def. 8.1.1 in [89]). *M runs in a polynomial time $O(n^k)$* if there exists some constant c such that M runs in at most cn^k steps for any input of length n, for $k, n \geq 0$.

3.3 Context-sensitive Languages

A strictly weaker language-generating model is represented by the *context-sensitive grammars* in which the form of the rewriting productions is more restrictive. Here, we define the aforementioned model formally.

Definition 3.31 (Context-sensitive grammar)**.** Let $G = (V, T, P, S)$ be an unrestricted grammar. G is a *context-sensitive* grammar if $p \in P$ implies $|\,\text{lhs}(p)| \leq |\,\text{rhs}(p)|$.

Definition 3.32 (Context-sensitive language)**.** A language, L, is *context-sensitive language* if there is a context-sensitive grammar, G, such that $L = L(G)$.

Definition 3.33 (Class of context-sensitive languages)**.** The *class of context-sensitive languages* is defined as $\mathbf{CS} = \{L : \ L = L(G) \text{ for an context-sensitive grammar } G\}$.

Definition 3.34 (Pentonnen normal form for context-sensitive grammars)**.** A context-sensitive grammar, $G = (V, T, P, S)$, is in *Pentonnen normal form* if every production, $p \in P$, has one of these three forms: $AB \rightarrow AC$, $A \rightarrow BC$, or $A \rightarrow a$ with $A, B, C \in V - T$ and $a \in T$.

3.4 Context-free Languages

From the practical viewpoint, the *context-free languages* represent the most fundamental part of the formal language theory. The form of rewriting productions is strictly prescribed by the constraints placed on their left-hand sides. Indeed, the left-hand side of a production must contain just one symbol. This section being the summary of context-free-rewriting-related definitions.

Definition 3.35 (Context-free grammar)**.** A *context-free grammar* is an quadruple $G = (V, T, P, S)$ where V is a total alphabet, T is an alphabet of *terminals* such that $T \subset V$, $P \subseteq (V - T) \times V^*$ is a finite binary relation, $S \in V - T$ is the start symbol.

Conventions

Hereafter, the members of P are called *productions*; accordingly, P is known as the *set of productions*. A production, $(A, x) \in P$, is customarily written as $A \rightarrow x$. For brevity, the productions in P are usually labelled; thus, a label p denotes a production $u \rightarrow v$ as $p : u \rightarrow v$ where u represents the left-hand side of p, denoted as $\text{lhs}(p)$, and v represents the right-hand side of p, denoted as $\text{rhs}(p)$. A production, p, is called as *ε-production* and *unit production* if $\text{rhs}(p) = \varepsilon$ and $\text{rhs}(p) \in V - T$, respectively. Whenever a notion *nonterminal* is used, it represents a member of $V - T$. For better readability, we often use the notation $A \rightarrow B|C \in P$ with the meaning $A \rightarrow B \in P$ and $A \rightarrow C \in P$.

Definition 3.36 (Direct derivation)**.** Let $G = (V, T, P, S)$ be a context-free grammar, $p \in P$, and $x, y \in V^*$. Then, $x\,\text{lhs}(p)y$ *directly derives* $x\,\text{rhs}(p)y$ according to p in G; denoted by $x\,\text{lhs}(p)y \Rightarrow x\,\text{rhs}(p)y\,[p]$ or, briefly, $x\,\text{lhs}(p)y \Rightarrow x\,\text{rhs}(p)y$.

Definition 3.37 (Derivation)**.** Let $G = (V, T, P, S)$ be a context-free grammar.

1. For any $u \in V^*$, G makes a *zero-step derivation* from u to u according to ε, which is written as $u \Rightarrow^0 u\,[\varepsilon]$,
2. Let $u_0, u_1, \dots, u_n \in V^*$, for some $n \geq 1$, such that $u_{i-1} \Rightarrow u_i\,[p_i]$ where $p_i \in P$, for $1 \leq i \leq n$; that is, $u_0 \Rightarrow u_1\,[p_1] \Rightarrow u_2\,[p_2] \Rightarrow \dots u_n\,[p_n]$. Then G makes an n-step derivation from u_0 to u_n according to $p_1 p_2 \dots p_n$, written as $u_0 \Rightarrow^n u_n\,[p_1 p_2 \dots p_n]$.

Let $v, w \in V^*$.

1. If there exists $n \geq 1$ so $v \Rightarrow^n w\,[\pi]$ in G, then v properly derives w according to π in G, written as $v \Rightarrow^+ w\,[\pi]$,
2. If there exists $n \geq 0$ so $v \Rightarrow^n w\,[\pi]$ in G, then v derives w according to π in G, written as $v \Rightarrow^* w\,[\pi]$.

Conventions

We frequently apply the simplify $v \Rightarrow^+ w\,[\pi]$ and $v \Rightarrow^* w\,[\pi]$ to $v \Rightarrow^+ w$ and $v \Rightarrow^* w$, respectively. If $S \Rightarrow^* w$ in G, then w is a *sentential form* of G. A sentential form, w, such that $w \in T^*$ is a *word* generated by G. A production, $p \in P$, is called *usable production* if there is a derivation $S \Rightarrow^* x\,\text{lhs}(p)y \Rightarrow x\,\text{rhs}(p)y \Rightarrow^* w$ in G, for $x, y \in V^*$ and $w \in T^*$.

Definition 3.38 (ε-free context-free grammar)**.** Let $G = (V, T, P, S)$ be a context-free grammar. G is an *ε-free context-free grammar* if $\text{rhs}(p) \neq \varepsilon$ for all $p \in P$.

Definition 3.39 (Generated language)**.** Let $G = (V, T, P, S)$ be a context-free grammar. The *language generated* by G, $L(G)$, is defined as $L(G) = \{w : \ w \in T^* \text{ and } S \Rightarrow^* w \text{ in } G\}$.

Definition 3.40 (Context-free language)**.** A language L is a *context-free language* if there exists a context-free grammar, G, such that $L = L(G)$.

Definition 3.41 (Classes of context-free languages and ε-free context-free languages)**.** The *class of context-free languages* is defined as $\mathbf{CF} = \{L : \ L = L(G) \text{ for a context-free grammar } G\}$. The *class of ε-free context-free* is defined as $\mathbf{CF}_\varepsilon = \{L : \ L = L(G) \text{ for an } \varepsilon\text{-free context-free grammar } G\}$.

Definition 3.42 (Leftmost direct derivation)**.** Let $G = (V, T, P, S)$ be a context-free grammar, $p \in P$, $x \in T^*$, and $y \in V^*$. Then, $x\,\text{lhs}(p)y$ *directly derives* $x\,\text{rhs}(p)y$ according to p in G in the *leftmost* way, as denoted by $x\,\text{lhs}(p)y \Rightarrow_{lm} x\,\text{rhs}(p)y\,[p]$ or, more briefly, by $x\,\text{lhs}(p)y \Rightarrow_{lm} x\,\text{rhs}(p)y$. Extend $\Rightarrow_{lm}$ to $\Rightarrow_{lm}^n$, $\Rightarrow_{lm}^+$, and $\Rightarrow_{lm}^*$ by analogy with the extension of $\Rightarrow$ to $\Rightarrow^n$, $\Rightarrow^+$, and $\Rightarrow^*$, respectively.

Derivation trees

Definition 3.43 (Production tree)**.** Let $G = (V, T, P, S)$ be a context-free grammar and $p \in P$. The *production tree*, $\text{pt}(p)$, corresponding to p is a labelled elementary tree, t, such that $\text{lhs}(p)$ labels $\text{root}(t)$ and $\text{fr}(t)$ is defined as follows:

1. If $|\,\text{rhs}(p)| = 0$ (that is, p is an ε-production), then $\text{fr}(t)$ consists of one node labelled by ε.
2. If $|\,\text{rhs}(p)| \geq 1$, then $\text{fr}(t)$ consists of $|\,\text{rhs}(p)|$ nodes that are labelled with the symbols appearing in $\text{rhs}(p)$ from left to right.

Definition 3.44 (Derivation tree)**.** Let $G = (V, T, P, S)$ be a context-free grammar. A *derivation tree* of G is a labelled tree, t, satisfying these two conditions:

1. $\text{root}(t)$ is labelled with a nonterminal $A \in V - T$.
2. Each elementary subtree t' appearing in t represents the production tree $\text{pt}(p)$ corresponding to a production, $p \in P$.

Definition 3.45 (Derivation tree corresponding to derivation)**.** Let $G = (V, T, P, S)$ be a context-free grammar. The correspondence between the derivation trees and the derivations that these trees represent is defined recursively as follows:

1. Let t be a one-node derivation tree such that $\text{root}(t)$ is labelled A, where $A \in V - T$. Then, t corresponds to $A \Rightarrow^* A\,[\varepsilon]$ in G.

2. Let t be the derivation tree corresponding to $A \Rightarrow^* x\,\mathrm{lhs}(p)y\,[\pi]$ in G. The derivation tree corresponding to $A \Rightarrow^* x\,\mathrm{lhs}(p)y\,[\pi] \Rightarrow x\,\mathrm{rhs}(p)y\,[p]$ is constructed by attaching $\mathrm{pt}(p)$ to the $|x| + 1$st leaf appearing in $\mathrm{fr}(t)$.

Definition 3.46 (Set of derivation trees)**.** Let $G = (V, T, P, S)$ be a context-free grammar. Then ${}_G\triangle(x)$ denote the set of all derivation trees of x in G with $x \in V^*$.

Definition 3.47 (Equivalent languages and grammars)**.** Let L_1 and L_2 be two context-free languages. If $L_1 - \{\varepsilon\} = L_2 - \{\varepsilon\}$ then L_1 and L_2 are *equivalent*, denoted by $L_1 = L_2$. If G_1 and G_2 are two context-free grammars such that $L(G_1) = L(G_2)$ then G_1 and G_2 are *equivalent.*

Definition 3.48 (Chomsky normal form for context-free grammars)**.** A context-free grammar, $G = (V, T, P, S)$, is in *Chomsky normal form* if every production, $p \in P$, satisfies $\mathrm{rhs}(p) \in T \cup (V - T)^2$.

Lemma 3.1 (Pumping lemma for context-free languages)**.** *Let L be a context-free language. Then, there exists a natural number, k, such that if $z \in L$ and $|z| \geq k$, then z can be expressed as $z = uvwxyz$ so that $vx \neq \varepsilon$; $|vwx| \leq k$; and $uv^nwx^my \in L$, for all $m \geq 0$.*

Ambiguity

Definition 3.49 (Ambiguity)**.** Let $G = (V, T, P, S)$ be a context-free grammar. If there exists a word $x \in L(G)$ such that $S \Rightarrow^*_{lm} x\,[\pi_1]$ and $S \Rightarrow^*_{lm} x\,[\pi_2]$ with $\pi_1 \neq \pi_2$, then G is *ambiguous*; otherwise, G is *unambiguous*.

Definition 3.50 (Bounded ambiguity)**.** Let $G = (V, T, P, S)$ be a context-free grammar. If for all words $x \in L(G)$ there is at most m, for some $m \geq 1$, such that $S \Rightarrow^*_{lm} x\,[\pi_1]$, $S \Rightarrow^*_{lm} x\,[\pi_2]$, ..., $S \Rightarrow^*_{lm} x\,[\pi_m]$ with $\pi_1 \neq \pi_2 \neq \cdots \neq \pi_m$, then G is *m-ambiguous*.

Definition 3.51 (Inherent ambiguity)**.** Let L be a context-free language. If every context-free grammar G satisfying $L(G) = L$ is ambiguous, then L is *inherently ambiguous*.

Pushdown Automata

Definition 3.52 (Pushdown automaton). A *pushdown automaton* is a septuple $M = (Q, \Sigma, \Gamma, R, q_0, Z_0, F)$, where Q is a finite set of states, Σ is an input alphabet, Γ is a pushdown alphabet, $R \subseteq (\Gamma \times Q \times (\Sigma \cup \{\varepsilon\})) \times (\Gamma^* \cup Q)$ is a finite binary relation, $q_0 \in Q$ is the start state, $Z_0 \in \Gamma$ is the initial pushdown symbol, $F \subseteq Q$ is a se of final states.

Conventions

For the conciseness, see Conventions in Sec. 3.6 for finite automata and apply them for pushdown automata analogously.

Definition 3.53 (Configuration)**.** Let $M = (Q, \Sigma, \Gamma, R, q_0, Z_0, F)$ be a pushdown automaton. A *configuration* of M is a word χ satisfying $\chi \in \Gamma^* Q \Sigma^*$.

Definition 3.54 (Move)**.** Let $M = (Q, \Sigma, \Gamma, R, q_0, Z_0, F)$ be a pushdown automaton. If $x\,\mathrm{lhs}(r)y$ is a configuration of M, where $x \in \Gamma^*$, $y \in \Sigma^*$, and $r \in R$, then M makes a move from $x\,\mathrm{lhs}(r)y$ to $x\,\mathrm{rhs}(r)y$ according to r, written as $x\,\mathrm{lhs}(r)y \vdash x\,\mathrm{rhs}(r)y\,[r]$. By analogy with the corresponding definition for a finite automata (see Def. 3.66), for all $n \geq 0$, define $\chi \vdash^n \chi'\,['\pi]$ in M, $\chi \vdash^+ \chi'\,[\pi]$, and $\chi' \vdash^* \chi'\,[\pi]$.

Definition 3.55 (Accepted language)**.** Let $M = (Q, \Sigma, \Gamma, R, q_0, Z_0, F)$ be a pushdown automaton and $w \in \Sigma^*$. The *language accepted by* M, denoted $L(M)$, is defined as $L(M) = \{w : w \in \Sigma^* \text{ and } Z_0 q_0 w \vdash^* f \text{ for some } f \in Q\}$.

3.5 Linear Languages

A much simpler than context-free language-describing model is represented by the notion of *linear grammars.* Indeed, during the rewriting process in a linear grammar, there is always at most one symbol that can be rewritten. The following definitions being the summary of the linear-rewriting-related definitions.

Definition 3.56 (Linear grammar)**.** Let $G = (V, T, P, S)$ be a context-free grammar. G is *linear grammar* if, for all $p \in P$, $\text{rhs}(p) \in T^*((V - T) \cup \{\varepsilon\})T^*$.

Definition 3.57 (Linear language)**.** Let L be a language. L is a *linear language* if there exists a linear grammar G such that $L(G) = L$.

Definition 3.58 (Class of linear languages)**.** The *class of linear languages* is defined as $\mathbf{LIN} = \{L : L = L(G) \text{ for a linear grammar } G\}$.

Lemma 3.2 (Pumping lemma for linear languages)**.** *Let L be a linear language. Then, there exists a natural number, k, such that if $z \in L$ and $|z| \geq k$, then z can be expressed as $z = uvwxyz$ so that $|vx| \geq 1$; $|uvxy| \leq k$; and $uv^n wx^m y \in L$, for all $m \geq 0$.*

3.6 Regular Languages

Regular languages were introduced as a finite characterization of infinite languages of the words in a specified format. In this section, we present several fundamental definitions related to the generating as well as the accepting regular languages.

Definition 3.59 (Regular expression)**.** Let Σ be an alphabet. The *regular expressions* over Σ and the languages that these expressions denote are defined recursively as: $\emptyset$ is a regular expression denoting the empty set; ε is a regular expression denoting $\{\varepsilon\}$; a, where $a \in \Sigma$ is a regular expression denoting $\{a\}$; if r and s are regular expressions denoting the languages R and S, respectively, then: $(r \cdot s)$ is a regular expression denoting RS; $(r + s)$ is a regular expression denoting $R \cup S$; and (r^*) is a regular expression denoting R^*.

Conventions

To simplify the *fully parenthesized regular expressions*, we reduce the number of parentheses in these expressions by assuming that * has a higher precedence than $\cdot$, and that $\cdot$ has a higher precedence than $+$. Furthermore, we abbreviate these expressions by omitting the symbol $\cdot$ in them. In addition, the expression rr^* is usually written as r^+ for brevity. For a regular expression r, $L(r)$ represents the language denoted by r.

Definition 3.60 (Regular grammar)**.** Let $G = (V, T, P, S)$ be a context-free grammar. G is *regular* if, for all $p \in P$, $\text{rhs}(p) \in T((V - T) \cup \{\varepsilon\})$.

Definition 3.61 (Regular language)**.** Let L be a language. L is a *regular language* if there exists a regular grammar G such that $L(G) = L$.

Definition 3.62 (Class of regular languages)**.** *Class of regular languages* is defined as $\mathbf{REG} = \{L : \; L = L(G) \text{ for a regular grammar } G\}$.

Definition 3.63 (Size of regular language)**.** Let R be a regular language. The *size* of R is denoted as $c(R)$ and defined as the number of states of a minimum-state finite automaton (see Def. 3.76) that accepts R. For any natural number $n \geq 1$, let $\mathbf{REG}_n$ be the class of all regular languages R such that $c(R) \leq n$.

Finite Automata

Definition 3.64 (Finite automaton)**.** A *finite automaton* is a quintuple $M = (Q, \Sigma, R, s, F)$ where Q is a finite set of states, Σ is an input alphabet such that $\Sigma \cap Q = \emptyset$, $R \subseteq (Q \times (\Sigma \cup \{\varepsilon\})) \times Q$ is a finite binary relation, $s \in Q$ is the start state, and $F \subseteq Q$ is a set of final states.

Conventions

Hereafter, members of R are called *computational rules* or, simply, *rules*; as a result, R is referred to as a *finite set of rules*. A rule, $(pa, q) \in R$ with $p, q \in Q$ and $a \in \Sigma \cup \{\varepsilon\}$, is usually written as $pa \rightarrow q$. For brevity, the rules of a finite automaton are usually labelled and these labels are used to refer to the rules. If the rule $pa \rightarrow q$ is labelled r, then we write $r : pa \rightarrow q$. Here, pa is the left-hand side of rule r, which is denoted by $\mathrm{lhs}(r)$; analogously, the right-hand side of rule r, i.e., q, is denoted by $\mathrm{rhs}(r)$.

Definition 3.65 (Configuration)**.** Let $M = (Q, \Sigma, R, s, F)$ be a finite automaton. A *configuration* of M is a word χ satisfying $\chi \in Q\Sigma^*$.

Definition 3.66 (Move)**.** Let $M = (Q, \Sigma, R, s, F)$ be a finite automaton. If $\mathrm{lhs}(r)y$ is a configuration of M, where $y \in \Sigma^*$ and $r \in R$, then M makes a move from $\mathrm{lhs}(r)y$ to $\mathrm{rhs}(y)$ according to r, written as $\mathrm{lhs}(r)y \vdash \mathrm{rhs}(y)y\,[r]$.

Conventions

When the specification of the rule r used in $\mathrm{lhs}(r)y \vdash \mathrm{rhs}(r)y\,[r]$ is immaterial, simply write $\mathrm{lhs}(r)y \vdash \mathrm{rhs}(r)y$. Given a finite automaton, M, χ_M denotes a configuration of M. When no confusion exists, we use χ instead of χ_M.

Definition 3.67 (Sequence of moves)**.** Let $M = (Q, \Sigma, R, s, F)$ be a finite automaton.

1. Let χ be any configuration of M. M makes *zero moves from* χ to χ *according to* ε, written as $\chi \vdash^0 \chi\,[\varepsilon]$.

2. Let there exist a sequence of configurations $\chi_0, \chi_1, \ldots, \chi_n$, for some $n \geq 1$ such that $\chi_{i-1} \vdash \chi_i \vdash [r_i]$, where $r_i \in R$, $1 \leq i \leq n$; that is, $\chi_0 \vdash \chi_1\,[r_1] \vdash \chi_2\,[r_2] \cdots \vdash \chi_n\,[r_n]$ Then, M makes n moves from χ_0 to χ_n according to $r_1 r_2 \ldots r_n$, written as $\chi_0 \vdash^n \chi_n\,[r_1 r_2 \ldots r_n]$.

Let χ and χ' be two configurations of M.

1. If there exists $n \geq 1$ so $\chi \vdash^n \chi'\,[\rho]$ in M, then $\chi \vdash^+ \chi'\,[\rho]$.

2. If there exists $n \geq 0$ so $\chi \vdash^n \chi'\,[\rho]$ in M, then $\chi \vdash^* \chi'\,[\rho]$.

Definition 3.68 (Accepted language)**.** Let $M = (Q, \Sigma, R, s, F)$ be a finite automaton and $w \in \Sigma^*$. The *language accepted by* M, denoted $L(M)$, is defined as $L(M) = \{w : w \in \Sigma^*$ and $sw \vdash^* f$ in M for some $f \in F\}$.

Definition 3.69 (ε-free finite automaton)**.** Let $M = (Q, \Sigma, R, s, F)$ be a finite automaton. M is an *ε-free finite automaton* if for all $r \in R$, $\text{lhs}(r) \in Q\Sigma$.

Definition 3.70 (Deterministic finite automaton)**.** Let $M = (Q, \Sigma, R, s, F)$ be an ε-free finite automaton such that for all $r, r' \in R$, $r \neq r'$ implies $\text{lhs}(r) \neq \text{lhs}(r')$. Then, M is a *deterministic finite automaton.*

Accessibility, Termination, and Completeness

Definition 3.71 (Accessible state)**.** Let $M = (Q, \Sigma, R, s, F)$ be finite automaton. A state $q \in Q$ is *accessible* if there exists a word w in Σ^* such that $sq \vdash^* q$ otherwise, q is *inaccessible.*

Definition 3.72 (Terminating state)**.** Let $M = (Q, \Sigma, R, s, F)$ be a deterministic finite automaton. A state $q \in Q$ is *termination* if there exists a word w in Σ^* such that $qw \vdash^* f$ for some $f \in F$; otherwise, q is *nonterminating.*

Definition 3.73 (Complete deterministic finite automaton)**.** A deterministic finite automaton $M = (Q, \Sigma, R, s, F)$ is *complete* if $\{\text{lhs}(r) : r \in R\} = Q\Sigma$ otherwise, M is *incomplete.*

Definition 3.74 (Well-specified finite automaton)**.** Let $M = (Q, \Sigma, R, s, F)$ be a complete deterministic finite automaton satisfying these two properties: Q has no inaccessible state,Q has no more than one nonterminating state. Then, M is a *well-specified* finite automaton.

Minimization

Definition 3.75 (Distinguishable states)**.** Let $M = (Q, \Sigma, R, s, F)$ be a well-specified finite automaton and $p, q \in Q$ so $p \neq q$. If there exists a word $w \in \Sigma^*$, so that $pw \vdash^* p'$ and $qw \vdash^* q'$ where $p', q' \in Q$ and $\text{card}(\{p', q'\} \cap F) = 1$, then w *distinguishes* p from q; at this point, p and q are *distinguishable.* If there exists no word that distinguishes p form p, p and q are *indistinguishable.*

Definition 3.76 (Minimum-state finite automaton)**.** Let M be a well-specified finite automaton. Let M_{min} be a well-specified finite automaton satisfying these two properties: M_{min} contains only distinguishable states; $L(M) = L(M_{min})$. Then, M_{min} is a *minimum-state finite automaton* equivalent to M.

3.7 Subregular Languages

In terms of the regulated rewriting theory, even the least powerful of the aforementioned models can be strong enough to increase the generative power of an underlying model (see Chap. 4 and Chap. 5). Thus, several even simpler models than regular grammars were introduced and they are defined in this section. For the definitions 3.77 through 3.85 consider a language, L, and the minimal alphabet V of L, $V = \text{alph}(L)$.

Definition 3.77 (Combinational language)**.** L is *combinational language* if and only if $L = V^*A$ for some $A \subseteq V$.

Definition 3.78 (Definite language)**.** L is *definite language* if and only if $L = A \cup V^*B$ where A and B are finite subsets of V^*.

Definition 3.79 (Nilpotent language)**.** L is *nilpotent language* if and only if L is finite or $V^* - L$ is finite.

Definition 3.80 (Commutative language)**.** L is *commutative language* if and only if $L = Comm(L)$ where $Comm(L) = \{a_{i_1}a_{i_2}\dots a_{i_n}\colon a_1a_2\dots a_n \in L,\ n \geq 1,\ \{i_1, i_2, \dots, i_n\} = \{1, 2, \dots, n\}\}$.

Definition 3.81 (Circular language)**.** L is *circular language* if and only if $L = Circ(L)$ where $Circ(L) = \{a_{i+1}a_{i+2}\dots a_na_1a_2\dots a_i\colon n \geq 1,\ 1 \leq i \leq n,\ a_1a_2\dots a_n \in L\}$.

Definition 3.82 (Suffix closed language)**.** L is *suffix closed language* if and only if $xy \in L$, for some words $x, y \in V^*$, implies $y \in L$ (or equivalently, $Suf(L) = L$, where $Suf(L) = \{y\colon xy \in L$ for some $x \in V^*\}$).

Definition 3.83 (Non-counting language)**.** L is *non-counting language* if and only if there is a natural number, $k \geq 1$, such that for any words $x, y, z \in V^*$, $xy^kz \in L$ if and only if $xy^{k+1}z \in L$.

Definition 3.84 (Power-separating language)**.** L is *power-separating language* if and only if for any $x \in V^*$ there is a natural number, $m \geq 1$, such that either $J_x^m \cap L = \emptyset$ or $J_x^m \subseteq L$ where $J_x^m = \{x^n\colon n \geq m\}$.

Definition 3.85 (Ordered language)**.** L is *ordered language* if and only if L is accepted by some finite automaton, $M = (Q, V, R, s, F)$, where $(Q, \preceq)$ is a totally ordered set and, for any $a \in V$, there are $q_1, q_2 \in Q$ such that $q_1 \preceq q_2$ implies $q_1' \preceq q_2'$ for some $q_1a \vdash q_1', q_2a \vdash q_2' \in R$, $q_1', q_2' \in Q$.

Definition 3.86 (Classes of combinational, definite, nilpotent, commutative, circular, suffix-closed, non-counting, power-separating, and ordered languages.)**.** The *class of combinational*, *definite*, *nilpotent*, *commutative*, *circular*, *suffix-closed*, *non-counting*, *power-separating*, and *ordered languages* are defined as **COMB**, **DEF**, **NIL**, **COM**, **CIRC**, **SUF**, **NC**, **PS**, **ORD** $= \{L$: L is a combinational, definite, nilpotent, commutative, circular, suffix-closed, non-counting, power-separating, and ordered language$\}$, respectively.

Definition 3.87 (Target sets of monoids)**.** Let V be an alphabet. Then, **MON** denote the class of all languages of the form V^*. For any natural number $n \geq 1$, let $\mathbf{MON}_n$ be the class of all languages that can be represented in the form $V_1^* \cup V_2^* \cup \dots \cup V_k^*$ with $1 \leq k \leq n$ where all V_i are alphabets, for $1 \leq i \leq k$.

3.8 Regulated Rewriting

This section being a brief summary of the notions related to the regulated rewriting theory which are required for the discussions in the following parts of this work. That is, we present several fundamental notions of the *matrix grammars* and the *scattered context grammars.*

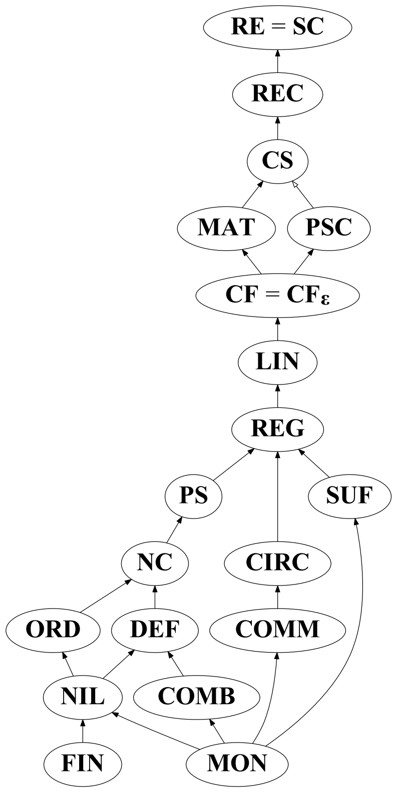

Figure 3.1: Hierarchy of language classes (a filled arrow from **X** to **Y** denotes $\mathbf{X} \subset \mathbf{Y}$, an empty arrow from **X** to **Y** denotes $\mathbf{X} \subseteq \mathbf{Y}$, and if two classes are not connected by a directed path then they are incomparable).

Matrix Grammar

Definition 3.88 (Matrix grammar)**.** A *matrix grammar* is a 5-tuple $G = (V, T, M, S)$ where V is a total alphabet, T is an alphabet of *terminals* such that $T \subset V$, M is a finite set of sequences of the form $m : (r_1, r_2. \dots, r_n)$, where $n \geq 1$, with the usual rewriting productions $r_i : A_i \rightarrow x_i$, where $A \in V - T$, $x \in V^*$, $1 \leq i \leq n$, $S \in V - T$ is the start symbol.

Conventions

Let $G = (V, T, M, S)$ be a matrix grammar. The elements of M are called matrices.

Definition 3.89 (Derivation)**.** Let $G = (V, T, M, S)$ be a matrix grammar. A derivation step in G is defined for $u, v \in V^*$ if and only if there are words $w_0, \dots, w_n \in V^*$, and a matrix $(r_1, r_2, \dots, r_n) \in M$, $r_i : A_i \rightarrow x_i$, $1 \leq i \leq n$, with $w_0 = u, w_n = v$ and $w_{j-1} = w'_{j-1} A_j w''_{j-1}, w_j = w'_{j-1} x_j w''_{j-1}$, for some $w'_{j-1}, w''_{j-1} \in V^*$, for all $1 \leq j \leq n$. By analogy with the corresponding definition for a context-free grammar (see Def. 3.37), for all $n \geq 0$, define $u \Rightarrow^n v$ in G. In a similar manner, introduce $u \Rightarrow^+ v$ and $u \Rightarrow^* v$.

Definition 3.90 (Generated language)**.** Let $G = (V, T, M, S)$ be a matrix grammar. The language generated by G, $L(G)$, is defined as $L(G) = \{x : \ x \in T^*, S \Rightarrow^* x\}$.

Definition 3.91 (Matrix language). A language L is a *matrix language* if there exists a matrix grammar, G, such that $L = L(G)$.

Definition 3.92 (Class of matrix languages). The *class of matrix languages* is defined as $\mathbf{MAT} = \{L : \ L = L(G) \text{ for a matrix grammar } G\}$.

Scattered Context Grammars

Definition 3.93 (Scattered context grammar). A *scattered context grammar* is a quadruple $G = (V, T, P, S)$ where V is a total alphabet, T is an alphabet of terminals such that $T \subset V$, P is a finite set of productions of the form $(A_1, A_2, \ldots, A_n) \to (x_1, x_2, \ldots, x_n)$ where $n \geq 1$, $A_i \in V - T$, and $x_i \in V^*$, for all $1 \leq i \leq n$, $S \in V - T$ is the start symbol.

Definition 3.94 (Derivation). Let $G = (V, T, P, S)$ be a scattered context grammar. If $u = u_1A_1 \ldots u_nA_nu_{n+1}$, $v = u_1x_1 \ldots u_nx_nu_{n+1}$ and $p = (A_1, \ldots, A_n) \to (x_1, \ldots, x_n) \in P$ where $u_i \in V^*$ for all $1 \leq i \leq n+1$ then G makes a derivation step from u to v according to p, written as $u {\Rightarrow} v\ [p]$ or simply $u {\Rightarrow} v$. By analogy with the corresponding definition for a context-free grammar (see Def. 3.37), for all $n \geq 0$, define $u {\Rightarrow}^n v$ in G. In a similar manner, introduce $u {\Rightarrow}^+ v$ and $u {\Rightarrow}^* v$.

Definition 3.95 (Generated language). Let $G = (V, T, P, S)$ be a scattered context grammar. The language generated by G, $L(G)$, is defined as $L(G) = \{x : \ x \in T^*, S {\Rightarrow}^* x\}$.

Definition 3.96 (Scattered context language). A language L is a *scattered context language* if there exists a scattered context grammar, G, such that $L = L(G)$.

Definition 3.97 (Class of scattered context languages). The *class of scattered context languages* is defined as $\mathbf{SC} = \{L : \ L = L(G) \text{ for a scattered context grammar } G\}$.

Definition 3.98 (Propagating scattered context grammar). A *propagating scattered context grammar* is a scattered context grammar $G = (V, T, P, S)$ in which each $(A_1, A_2, \ldots, A_n) \to (x_1, x_2, \ldots, x_n) \in P$ satisfies $x_i \in V^+$, for all $1 \leq i \leq n$.

Definition 3.99 (Propagating scattered context language). A language L is a *propagating scattered context language* if there exists a propagating scattered context grammar, G, such that $L = L(G)$.

Definition 3.100 (Class of propagating scattered context languages). The *class of propagating scattered context languages* is defined as $\mathbf{PSC} = \{L : \ L = L(G) \text{ for a propagating scattered context grammar } G\}$.

3.9 Language Classes Hierarchy

Through whole this chapter, we have presented several fundamental language-describing models that differ namely in their generative power. The relations between the language classes aforementioned in this chapter can be given by Fig. 3.1 (see [48] and [89]). For brevity, some of the definitions common for all presented grammars are pointed out here.

Definition 3.101 (Set of all grammars of specified type). The set of all regular, linear, ε-free context-free, context-free, context-sensitive, and unrestricted grammars is denoted as $\mathbb{G}_{\mathbf{REG}}$, $\mathbb{G}_{\mathbf{LIN}}$, $\mathbb{G}_{\mathbf{CF}_\varepsilon}$, $\mathbb{G}_{\mathbf{CF}}$, $\mathbb{G}_{\mathbf{CS}}$, and $\mathbb{G}_{\mathbf{RE}}$, respectively.

Conventions

If a type of a grammar represents an immaterial piece of information, it is omitted in these notations.

Consider a language class $\mathbf{X}$ and a language operation o. If $\mathbf{X}$ contains every language resulting from the application of o to any language in $\mathbf{X}$, then $\mathbf{X}$ is *closed* under o; otherwise, L is not closed under o.

3.10 Transducers

Except of the generating and accepting the formal languages, we often deal with the notion of *translating* one word into another or one language into another. There are several models used for the translating the languages. Two of the most fundamentals are known as *generalized sequential machine mapping* and *rational transduction.*

Generalized Sequential Machine Mapping

Definition 3.102 (Generalized sequential machine mapping)**.** A *generalized sequential machine* is a 6-tuple $M = (Q, \Sigma, \Omega, \tau, s, F)$ where Q is finite set of states, Σ is an input alphabet, Ω is an output alphabet, τ is a finite subset of $(S \times \Sigma) \times (S \times \Omega^*)$ called transition function, $s \in Q$ is initial state, $F \subseteq Q$ is a finite set of final states.

Definition 3.103 (Configuration)**.** Let $M = (Q, \Sigma, \Omega, \tau, s, F)$ be a generalized sequential machine. A configuration of M is (p, u) with $p \in Q$, $u \in \Sigma^*$. A configuration (p, u) is initial, final if $p = s$, $p \in F$, respectively.

Definition 3.104 (Generalized sequential machine mapping of word)**.** Let $M = (Q, \Sigma, \Omega, \tau, s, F)$ be a generalized sequential machine. For every input word $u \in \Sigma^*$, *generalized sequential machine mapping* of u, *gsm mapping* for short, is denoted as $GSM_M(u)$ and defined as $GSM_M(u) = \{v \in \Omega^* : (t, v) \in \tau(s, u)$ is a final configuration$\}$.

Definition 3.105 (Generalized sequential machine mapping of language)**.** Let $M = (Q, \Sigma, \Omega, \tau, s, F)$ be a generalized sequential machine. A *generalized sequential machine mapping* of a language L, *gsm mapping* for short, is denoted as $GSM_M(L)$ and defined as $GSM_M(L) = \bigcup_{u \in L} GSM_M(u)$.

Rational Transduction

Definition 3.106 (Rational transducer)**.** A *rational transducer* is a 6-tuple $M = (Q, \Sigma, \Omega, \tau, s, F)$ where Q is finite set of states, Σ is an input alphabet, Ω is an output alphabet, τ is a finite subset of $(S \times \Sigma^*) \times (S \times \Omega^*)$ called transition function, $s \in Q$ is initial state, $F \subseteq Q$ is a finite set of final states.

Definition 3.107 (Configuration)**.** Let $M = (Q, \Sigma, \Omega, \tau, s, F)$ be a rational transducer. A configuration of M is (p, u) with $p \in Q$, $u \in \Sigma^*$. A configuration (p, u) is initial, final if $p = s$, $p \in F$, respectively.

Definition 3.108 (Rational transduction of word)**.** Let $M = (Q, \Sigma, \Omega, \tau, s, F)$ be a rational transducer. For every input word $u \in \Sigma^*$, *rational transduction* of u is denoted as $RT_M(u)$ and defined as $RT_M(u) = \{v \in \Omega^* : (t, v) \in \tau(s, u)$ is a final configuration$\}$.

Definition 3.109 (Rational transduction of language)**.** Let $M = (Q, \Sigma, \Omega, \tau, s, F)$ be a rational transducer. A *rational transduction* of a language L is denoted as $RT_M(L)$ and defined as $RT_M(L) = \bigcup_{u \in L} RT_M(u)$.

Bibliographical Notes

As a reference literature for both the basic definitions as well as the notation, we use [89] wherever it is possible. For context-free languages, we also use [6], [7], [19], [20], [39], [44], and [96]. Regular languages and finite automata are also covered in [13], [40], [51], [55], [59], [86], [87], [121], and [133]. Pumping lemmata in more detail can be studied from [9], [14], [21], [35], [47], [52], [132], [135], and [137]. Other formal-language-theory-related defninitons are derived from [2], [8], [27], [28], [45], [51], [58], [71], [82], [99], [100], [103], [118], [120], [131], and [136]. The relations between the classes of subregular languages are stuided in [48]. Furhter properties of rational tranducers can be found in [85]. The definitons of orderings are derived from [110]. As a basic underlying literature for regulated models, we use [26], more regulated-rewriting-related studies can be found in [32], [33], [36], [91], [93], [113], [115], [129], [138], and many others. For more about matrix grammars and their properties see [1], several modificication of matrix grammars can be found in [95], [101], [102] [114], [127], and [128]. The summary of known results related to scattered context grammars is given in [92], we point out several other publication related to scattered context rewriting and its modification, namely [46] where scattered context grammars are introduced and also [23], [30], [37], [38], [83], [84], [94], and [126].

Part II

State of the Art Survey

Chapter 4

Tree Controlled Grammars

This chapter being the formal description of level-based restriction placed upon the derivation trees as it was informally presented in the introduction of this work (see Sec. 1.1.1).

4.1 Definitions

In Sec. 1.1.1, *tree controlled grammars* are described informally. Here, we present the corresponding strictly formal definitions.

Definition 4.1 (Tree controlled grammar)**.** A *tree controlled grammar* is a pair (G, R) where $G = (V, T, P, S)$ is a controlled grammar and $R \subseteq V^*$ is a control language.

Definition 4.2 (Tree controlled language)**.** Let (G, R) be a tree controlled grammar. The *language generated by* (G, R) is denoted by $L(G, R)$ and defined by

$$\begin{aligned} L(G, R) = \{x : {} & x \in L(G) \text{ and there exists a derivation tree of } x \text{ in } G \text{ such that} \\ & \text{each word obtained by concatenation of all symbols at any level} \\ & \text{of } t \text{ (except the last one) from left to right is in } R\}. \end{aligned}$$

Definition 4.3 (Class of tree controlled languages)**.** For $\mathbf{X} \in \{\mathbf{CF}, \mathbf{CF}_\varepsilon, \mathbf{REG}\}$ and $\mathbf{Y} \in \{\mathbf{RE}, \mathbf{CS}, \mathbf{CF}, \mathbf{REG}, \mathbf{FIN}\} \cup \{\mathbf{MON}, \mathbf{NIL}, \mathbf{COMB}, \mathbf{ORD}, \mathbf{DEF}, \mathbf{COM}, \mathbf{NC}, \mathbf{CIRC}, \mathbf{PS}, \mathbf{SUF}\} \cup \{\mathbf{MON}_n, \mathbf{REG}_n : n \geq 1\}$, the *class of tree controlled languages* is denoted as $\mathbf{TC}(\mathbf{X}, \mathbf{Y})$ and defined as

$$\begin{aligned} \mathbf{TC}(\mathbf{X}, \mathbf{Y}) = \{L(G, R) : {} & (G, R) \text{ is a tree controlled grammar in which } G \in \mathbb{G}_{\mathbf{X}} \\ & \text{and } R \in \mathbf{Y}\}. \end{aligned}$$

Definition 4.4 (Nonterminal complexity of tree controlled grammar)**.** Let (G, R) be a tree controlled grammar. Clearly, $R = L(G')$ for some $G' = (V', T', P', S')$. A *nonterminal complexity of tree controlled grammar* is denoted as $\mathrm{Var}(G, R)$ and defined as $\mathrm{Var}(G, R) = \mathrm{card}(V - T) + \mathrm{card}(V' - T')$.

Definition 4.5 (Nonterminal complexity of tree controlled language)**.** Let $\min(\Sigma)$ denote the smallest number in a given set of natural numbers, Σ. Let $L \in \mathbf{TC}(\mathbf{CF}_\varepsilon, \mathbf{REG})$. An *$\varepsilon$-free nonterminal complexity of tree controlled language* is denoted as $\mathrm{Var}_\varepsilon(L)$ and defined as

$$\begin{aligned} \mathrm{Var}_\varepsilon(L) = \min(\{\mathrm{Var}(G, R) : {} & (G, R) \text{ is a tree controlled grammar with an } \varepsilon\text{-free context-} \\ & \text{free grammar } G \text{, regular language } R \text{, and } L(G, R) = L\}). \end{aligned}$$

Let $L \in \mathbf{TC}(\mathbf{CF}, \mathbf{REG})$. A *nonterminal complexity of tree controlled language* is denoted as $\mathrm{Var}(L)$ and defined as

$$\mathrm{Var}(L) = \min(\{\mathrm{Var}(G,R) : (G,R) \text{ is a tree controlled grammar with a context-free grammar } G\text{, regular language } R\text{, and } L(G,R) = L\}).$$

4.2 Examples

The following three examples illustrate the definitions of tree controlled grammars introduced in the previous section as well as the languages they generate.

Example 4.1. Consider tree controlled grammar (G, R) where

$$\begin{aligned} G &= (\{S, a\}, \{a\}, P, S),\\ P &= \{S \to a, \quad S \to SS\},\\ R &= \{S^* : \ n \geq 0\}. \end{aligned}$$

Clearly, $L(G,R) = \{a^{2^n} : \ n \geq 0\} \notin \mathbf{CF}$ and $L(G,R) \in \mathbf{TC}(\mathbf{CF}, \mathbf{REG})$.

Example 4.2. Consider tree controlled grammar (G, R) where

$$\begin{aligned} G &= (\{S, A, B, C, a, b, c\}, \{a, b, c\}, P, S),\\ P &= \{S \to ABC, \quad A \to aA, \quad A \to a, \quad B \to bB, \quad B \to b, \quad C \to cC, \quad C \to c\},\\ R &= \{S, ABC, aAbBcC\}. \end{aligned}$$

Clearly, $L(G,R) = \{a^n b^n c^n : \ n \geq 1\} \notin \mathbf{CF}$ and $L(G,R) \in \mathbf{TC}(\mathbf{CF}, \mathbf{FIN})$.

Example 4.3. Consider tree controlled grammar (G, R) where

$$\begin{aligned} G &= (\{S, A, B, C, D, E, a, b\}, \{a, b\}, P, S),\\ P &= \{S \to AB, \quad A \to aAb, \quad B \to Ba, \quad A \to ab,\\ &\qquad B \to a, \quad A \to aCb, \quad C \to Cb, \quad C \to c\},\\ R &= \{S, AB, aAbBa, aCba, Cb\}. \end{aligned}$$

Clearly, $L(G,R) = \{a^n b^{n+m} a^n : \ n \geq 1, m \geq 0\} \notin \mathbf{CF}$ and $L(G,R) \in \mathbf{TC}(\mathbf{CF}, \mathbf{FIN})$.

4.3 Results

The results presented in this section represent the state of the art of level-based restriction placed upon the derivation trees investigation area. For the informal description of the corresponding state of the art, see Sec. 1.1.1.

Theorem 4.1 (Th. 3.1 in [24]). *If (G, R) is a tree controlled grammar where G is unambiguous context-free grammar and R is a regular language, then there exists an algorithm which for any word, w, with $|w| = n$ determines in $O(n^2)$ steps if $w \in L(G,R)$.*

Theorem 4.2 (Th. 3.2 in [24]). *For every tree controlled grammar, (G, R) with ε-free context-free grammar, $G = (V, T, P, S)$, and regular language, R, the language $L(G,R)$ is recursive.*

Theorem 4.3 (Th. 3.3 in [24]). *A language, L, is regular, linear, context-free if and only if there is a context-free grammar, $G = (V, T, P, S)$, such that $L = L(G, T^*(V - T))$, $L = L(G, T^*(V - T)T^*)$, $L = L(G, V^*)$, respectively.*

Theorem 4.4 (Th. 3.5 in [24]). *Let Σ be an alphabet. Then, there exists a context-free grammar, $G = (V, \Sigma, P, S)$, such that for each recursively enumerable language, $L \subseteq \Sigma^*$, there exists a regular language, $R \subseteq V^*$, such that $L = L(G, R)$.*

Corollary 4.5 (Col. 3.1 in [24]). *Every recursively enumerable language can be generated by a tree controlled grammar, (G, R), such that $G = (V, \Sigma, P, S)$ is a context-free grammar, R is a regular language, and $P \subseteq (V - \Sigma) \times ((V - \Sigma)^* \cup \Sigma)$.*

Theorem 4.6 (Th. 3.6 in [24]). *Every recursively enumerable language can be generated by a tree controlled grammar, (G, R), with a context-free grammar, $G = (V, T, P, S)$, and a regular language, $R \subseteq (V - T)^*$.*

Theorem 4.7 (Th. 1 in [104]). *If G is a regular grammar, then for all $R \in \mathbf{X}$, where $\mathbf{X} \in \{\mathbf{RE}, \mathbf{CS}, \mathbf{CF}, \mathbf{REG}, \mathbf{FIN}\}$, $L(G, R) \in \mathbf{REG}$.*

Corollary 4.8 (Col. in Sec. 2 in [104]).

For all $\mathbf{X} \in \{\mathbf{RE}, \mathbf{CS}, \mathbf{CF}, \mathbf{REG}, \mathbf{FIN}\}$, $\mathbf{TC}(\mathbf{REG}, \mathbf{X}) = \mathbf{REG}$.

Theorem 4.9 (Th. 2 in [104]). $\mathbf{TC}(\mathbf{CF}_\varepsilon, \mathbf{CS}) \subseteq \mathbf{CS}$.

Theorem 4.10 (Th. 3 in [104]). $\mathbf{X} \subseteq \mathbf{TC}(\mathbf{CF}_\varepsilon, \mathbf{X})$, *for* $\mathbf{X} \in \{\mathbf{RE}, \mathbf{CS}, \mathbf{CF}, \mathbf{REG}\}$.

Corollary 4.11 (Col. in Sec. 2 in [104]). $\mathbf{TC}(\mathbf{CF}_\varepsilon, \mathbf{RE}) = \mathbf{RE}$ *and* $\mathbf{TC}(\mathbf{CF}_\varepsilon, \mathbf{CS}) = \mathbf{CS}$.

Theorem 4.12 (Th. 4 in [104]). $\mathbf{CS} \subseteq \mathbf{TC}(\mathbf{CF}_\varepsilon, \mathbf{REG})$.

Corollary 4.13 (Col. in Sec. 2 in [104]). $\mathbf{TC}(\mathbf{CF}_\varepsilon, \mathbf{X}) = \mathbf{CS}$, *for* $\mathbf{X} \in \{\mathbf{CS}, \mathbf{CF}, \mathbf{REG}\}$.

Theorem 4.14 (Th. 5 in [104]). $\mathbf{TC}(\mathbf{X}, \mathbf{FIN}) = \mathbf{FIN}$, *for* $\mathbf{X} \in \{\mathbf{CF}, \mathbf{CF}_\varepsilon\}$.

Corollary 4.15 (Col. in Sec. 2 in [104]).

Any language over one-letter alphabet in $\mathbf{TC}(\mathbf{CF}, \mathbf{FIN})$ *is regular.*

Lemma 4.16 (Lem. 4 in [29]). *If* $\mathbf{X} \subseteq \mathbf{Y} \subseteq \mathbf{REG}$, *then* $\mathbf{TC}(\mathbf{CF}_\varepsilon, \mathbf{X}) \subseteq \mathbf{TC}(\mathbf{CF}_\varepsilon, \mathbf{Y})$, *for* $\mathbf{X}, \mathbf{Y} \in \{\mathbf{FIN}, \mathbf{MON}, \mathbf{NIL}, \mathbf{COMB}, \mathbf{ORD}, \mathbf{DEF}, \mathbf{COM}, \mathbf{NC}, \mathbf{CIRC}, \mathbf{PS}, \mathbf{SUF}\}$.

Theorem 4.17 (Th. 5 in [29]). $\mathbf{TC}(\mathbf{CF}_\varepsilon, \mathbf{SUF}) = \mathbf{CS}$.

Theorem 4.18 (Th. 6 in [29]). $\mathbf{TC}(\mathbf{CF}_\varepsilon, \mathbf{CIRC}) = \mathbf{CS}$.

Theorem 4.19 (Th. 7 in [29]). $\mathbf{TC}(\mathbf{CF}_\varepsilon, \mathbf{ORD}) = \mathbf{CS}$.

Corollary 4.20 (Col. 8 in [29]). $\mathbf{TC}(\mathbf{CF}_\varepsilon, \mathbf{NC}) = \mathbf{TC}(\mathbf{CF}_\varepsilon, \mathbf{PS}) = \mathbf{CS}$.

Theorem 4.21 (Th. 9 in [29]). $\mathbf{TC}(\mathbf{CF}_\varepsilon, \mathbf{COM}) = \mathbf{MAT}_\varepsilon$ *where* $\mathbf{MAT}_\varepsilon$ *denotes the class of languages generated by matrix grammars in which all matrices contains no erasing productions.*

Theorem 4.22 (Th. 10 in [29]). $\mathbf{TC}(\mathbf{CF}_\varepsilon, \mathbf{FIN}) = \mathbf{MAT}_{fin}$ *where* $\mathbf{MAT}_{fin}$ *denotes the class of languages generated by matrix grammars of finite index in which all matrices contains no erasing productions.*

Theorem 4.23 (Th. 12 in [29]).

$\mathbf{TC}(\mathbf{CF}_\varepsilon, \mathbf{FIN}) \subset \mathbf{TC}(\mathbf{CF}_\varepsilon, \mathbf{NIL})$ *and*
$\mathbf{TC}(\mathbf{CF}_\varepsilon, \mathbf{MON}) \subset \mathbf{TC}(\mathbf{NIL})$.

Theorem 4.24 (Th. 13 in [29]).

$\mathbf{RE} = \mathbf{TC}(\mathbf{CF}, \mathbf{REG}) = \mathbf{TC}(\mathbf{CF}, \mathbf{SUF}) = \mathbf{TC}(\mathbf{CF}, \mathbf{ORD}) = \mathbf{TC}(\mathbf{CF}, \mathbf{NC}) =$
$= \mathbf{TC}(\mathbf{CF}, \mathbf{PS}) = \mathbf{TC}(\mathbf{CF}, \mathbf{COM}) = \mathbf{TC}(\mathbf{CF}, \mathbf{CIRC})$,
$\mathbf{MAT}_{fin} = \mathbf{TC}(\mathbf{CF}, \mathbf{FIN}) \subset \mathbf{TC}(\mathbf{CF}, \mathbf{NIL}) \subseteq \mathbf{RE}$,
$\mathbf{TC}(\mathbf{CF}, \mathbf{MON}) \subset \mathbf{TC}(\mathbf{CF}, \mathbf{NIL})$, *and* $\mathbf{TC}(\mathbf{CF}, \mathbf{MON}) \subset \mathbf{TC}(\mathbf{CF}, \mathbf{DEF})$.

Theorem 4.25 (Prop. 4 in [28]).

$\mathbf{TC}(\mathbf{CF}_\varepsilon, \mathbf{MON}_1) \subseteq \mathbf{TC}(\mathbf{CF}_\varepsilon, \mathbf{MON}_2) \subseteq \cdots \subseteq \mathbf{TC}(\mathbf{CF}_\varepsilon, \mathbf{MON}_j) \subseteq \ldots$.

Lemma 4.26 (Lem. 10 in [28]).

$\mathbf{TC}(\mathbf{CF}_\varepsilon, \mathbf{REG}_1) \subseteq \mathbf{TC}(\mathbf{CF}_\varepsilon, \mathbf{REG}_2) \ldots \mathbf{TC}(\mathbf{CF}_\varepsilon, \mathbf{REG}_n) \subseteq \ldots$.

Theorem 4.27 (Th. 12 in [28]). $\mathbf{TC}(\mathbf{CF}_\varepsilon, \mathbf{REG}_1) \subset \mathbf{TC}(\mathbf{CF}_\varepsilon, \mathbf{REG}_2)$.

Theorem 4.28 (Th. 13 in [28]). $\mathbf{TC}(\mathbf{CF}_\varepsilon, \mathbf{COMB}) \subseteq \mathbf{TC}(\mathbf{CF}_\varepsilon, \mathbf{REG}_2)$.

Theorem 4.29 (Th. 18 in [28]). *Every language that is generated by a context-sensitive grammar with exactly r non-context-free productions $p_1, p_2, \ldots, p_r$ and n_i symbols on the left-hand side of the production p_i, for $1 \leq i \leq r$, is also generated by a tree controlled grammar where the control set is accepted by a deterministic finite automaton with at most $2 + \Sigma_{i=1}^{r}(n_i - 1)$ states.*

Lemma 4.30 (Lem. 5 in [118]). *For every linearly bounded queue automaton (see Def. 3 in [118] for the definition), A, there is a tree controlled grammar, (G, R), such that $L(G, R) = L(A)$.*

Theorem 4.31 (Th. 6 in [118]). $\mathbf{TC}(\mathbf{CF}_\varepsilon, \mathbf{REG}_5) = \mathbf{CS}$.

Theorem 4.32 (Th. 7 in [118]). $\mathbf{TC}(\mathbf{CF}, \mathbf{REG}_5) = \mathbf{RE}$.

Theorem 4.33 (Th. 4 in [123]). *Every recursively enumerable language can be generated by a tree controlled grammar, (G, R), with context-free grammar, G, and regular language, R, such that* $\mathrm{Var}(G, R) \leq 9$.

Theorem 4.34 (Th. 1 in [122]). *Every recursively enumerable language can be generated by a tree controlled grammar, (G, R), with context-free grammar, G, and regular language, R, such that* $\mathrm{Var}(G, R) \leq 7$.

Theorem 4.35 (Th. 2 in [122]). *For any regular language, L, there is a tree controlled grammar, (G, R), with context-free grammar, G, and regular language, R, such that $L(G, R) = L$ and* $\mathrm{Var}(G, R) = 3$.

Theorem 4.36 (Th. 4 in [122]). *For any regular simple matrix grammar, G, (see [101] for the definition) there is a tree controlled grammar, (G, R), with context-free grammar, G, and regular language, R, such that $L(G, R) = L(G)$ and* $\mathrm{Var}(G, R) = 3$.

Theorem 4.37 (Th. 5 in [122]). *For any linear language, L, there is a tree controlled grammar, (G, R), such that $L(G, R) = L$ and* $\mathrm{Var}(G, R) = 3$.

Lemma 4.38 (Lem. 2 in [123]). *For $n \geq 1$, let $L_n = \bigcup_{i=1}^{n}\{a_i^j : j \geq 1\}$. Then,* $\mathrm{Var}_\varepsilon(L_n) = n + 1$.

Chapter 5

Path Controlled Grammars

In this chapter, based on the informal survey introduced in Sec. 1.1.2, we present the definitions and the results related to path controlled grammar.

5.1 Definitions

The following definitions are needed for presenting the state of the art in path-based restriction placed upon the derivation trees investigation area.

Definition 5.1 (Language of the paths)**.** Let $G = (V, T, P, S)$ be a context-free grammar and $t \in {}_G\triangle(w)$ for some $w \in L(G)$. Then,

$$\begin{aligned}&\mathit{path}(t) = \{\text{word}(p) : \ p \text{ is a path of } t\},\\ &\mathit{path}({}_G\triangle(w)) = \bigcup\{\mathit{path}(t) : \ t \in {}_G\triangle(w)\}, \text{ and}\\ &\mathit{path}(G) = \bigcup\{\mathit{path}({}_G\triangle(w)) : \text{for all } w \in L(G)\}.\end{aligned}$$

Definition 5.2 (Path controlled grammar)**.** A path controlled grammar is a pair (G, G') where $G = (V, T, P, S)$ is a controlled grammar and $G' = (V', V, S', P')$ is a controlling grammar.

Definition 5.3 (Path controlled language)**.** Let (G, G') be a path controlled grammar. The language generated by (G, G') is denoted by $L(G, G')$ and defined by

$$L(G, G') = \{w \in L(G) : \ \mathit{path}(w) \cap L(G') \neq \emptyset\}.$$

Definition 5.4 (Class of path controlled languages)**.** For $\mathbf{X}, \mathbf{Y} \in \{\mathbf{CF}, \mathbf{LIN}, \mathbf{REG}\}$, the *class of path controlled languages* is denoted as $\mathbf{PC}(\mathbf{X}, \mathbf{Y})$ and defined as

$$\begin{aligned}\mathbf{PC}(\mathbf{X}, \mathbf{Y}) = \{L(G, G') : \ &(G, G') \text{ is a path controlled grammar in which } G \in \mathbb{G}_{\mathbf{X}} \text{ and}\\ &G' \in \mathbb{G}_{\mathbf{Y}}\}.\end{aligned}$$

5.2 Examples

The following two examples illustrate the definitions of path controlled grammars introduced in the previous section as well as the languages they generate.

Example 5.1. Consider path controlled grammar (G, G') where

$G = (\{S, B, D, a, b, c, d\}, \{a, b, c, d\}, P, S)$,
$P = \{S \to aSd, \quad S \to aBd, \quad B \to bBc, \quad B \to D, \quad D \to bc\}$,
$L(G') = \{S^n B^n Db : \; n \geq 1\}$.

Clearly, $L(G, G') = \{a^k b^k c^k d^k : \; k \geq 1\} \notin \mathbf{CF}$ and $L(G, G') \in \mathbf{PC}(\mathbf{LIN}, \mathbf{LIN})$.

Example 5.2. Consider path controlled grammar (G, G') where

$G = (\{S, A, B, C, D, a, b\}, \{a, b\}, P, S)$,
$P = \{S \to aS, \quad S \to aB, \quad B \to Bb, \quad B \to A, \quad A \to bA,$
$\quad A \to C, \quad C \to Ca, \quad C \to D, \quad D \to a\}$,
$L(G') = \{S^n B^m A^m C^n Da : \; n, m \geq 1\}$.

Clearly, $L(G, G') = \{a^k b^l a^k b^l : \; k, l \geq 1\} \notin \mathbf{CF}$ and $L(G, G') \in \mathbf{PC}(\mathbf{LIN}, \mathbf{LIN})$.

5.3 Results

This section being the summary of the fundamentals in the state of the art of path-based restriction placed upon the derivation trees investigation area. For the informal description of the corresponding state of the art, see Sec. 1.1.2.

Theorem 5.1 (Prop. 1 in [80]). *If G is a context-free grammar, then path$(G) \in \mathbf{REG}$.*

Theorem 5.2 (Prop. 2 in [80]). $\mathbf{X} = \mathbf{PC}(\mathbf{X}, \mathbf{REG})$*, for all* $\mathbf{X} \in \{\mathbf{REG}, \mathbf{LIN}, \mathbf{CF}\}$.

Theorem 5.3 (Prop. 3 in [80]). $\mathbf{PC}(\mathbf{REG}, \mathbf{X}) \subseteq \mathbf{X}$*, for all* $\mathbf{X} \in \{\mathbf{LIN}, \mathbf{CF}\}$.

Theorem 5.4 (Prop. 4 in [80]). *If L is a language in $\mathbf{X} \in \{\mathbf{LIN}, \mathbf{CF}\}$ without words of length one, then $L \in \mathbf{PC}(\mathbf{REG}, \mathbf{X})$.*

Theorem 5.5 (Prop. 6 in [80]; see the discussion in Sec. 8.3.4). $\mathbf{PC}(\mathbf{CF}, \mathbf{CF}) \subseteq \mathbf{MAT}$.

Theorem 5.6 (Prop. 7 in [80]). *If $L \subseteq V^*$, $L \in \mathbf{PC}(\mathbf{CF}, \mathbf{CF})$, then there are two constants, p and q, such that each word, $z \in L$, with $|z| > p$ can be written in the form $z = u_1v_1u_2v_2u_3v_3u_4v_4u_5$, such that $0 < |v_1v_2v_3v_4| \leq q$ and $u_1v_1^iu_2v_2^iu_3v_3^iu_4v_4^iu_5 \in L$ for all $i \geq 1$.*

Theorem 5.7 (Prop. 8 in [80]). *If $L \subseteq V^*$, $L \in \mathbf{PC}(\mathbf{LIN}, \mathbf{LIN})$, then there are two constants, p and q, such that each word, $z \in L$, with $|z| > p$ can be written in the form $z = u_1v_1u_2v_2u_3v_3u_4v_4u_5$, such that $0 < |v_1v_2v_3v_4| \leq q$, $|u_1u_2u_3u_4| \leq q$, and $u_1v_1^iu_2v_2^iu_3v_3^iu_4v_4^iu_5 \in L$ for all $i \geq 1$.*

Corollary 5.8 (Conseq. in [80]; see the discussion in Sec. 8.3.4). $\mathbf{PC}(\mathbf{CF}, \mathbf{CF}) \subset \mathbf{MAT}$.

Corollary 5.9 (Conseq. in [80]). $\mathbf{PC}(\mathbf{LIN}, \mathbf{LIN})$ *is not closed under concatenation.*

Theorem 5.10 (Prop. 9 in [80]). *For each language, $L \subseteq V^*$, $L \in \mathbf{PC}(\mathbf{LIN}, \mathbf{LIN})$, there are three linear languages $L_1 \subseteq V^*\{c\}V^*$, $L_2 = V^*\{c\}V^*$, and $L_3 = V^*$, where $c \notin V$, such that:*

- $L \subseteq \{u_1u_2u_3u_4u_5 | \; u_1cu_5 \in L_1, u_2cu_4 \in L_2, u_3 \in L_3\}$.

- *For each word, $u_1cu_5 \in L_1$ (for each word, $u_2cu_4 \in L_2$, for each word, $u_3 \in L_3$), there are a word, $u_2cu_4 \in L_2$, and a word, $u_3 \in L_3$ (a word, $u_1cu_5 \in L_1$, and a word, $u_3 \in L_3$, respectively, a word, $u_1cu_5 \in L_1$, and a word, $u_2cu_4 \in L_2$) such that $u_1u_2u_3u_4u_5 \in L$.*

Theorem 5.11 (Prop. 10 in [80])**.**

$\mathbf{CF} - \mathbf{PC}(\mathbf{LIN}, \mathbf{LIN}) \neq \emptyset$.
The inclusion $\mathbf{PC}(\mathbf{LIN}, \mathbf{LIN}) \subset \mathbf{PC}(\mathbf{CF}, \mathbf{CF})$ is proper.

Theorem 5.12 (Prop. 11 in [80])**.** *If (G, G') is a path controlled grammar with linear grammars G and G' such that G has a bounded ambiguity, the parsing of words in $L(G, G')$ can be done in a polynomial time.*

Theorem 5.13 (Prop. 1 in [81])**.** $\mathbf{PC}(\mathbf{CF}, \mathbf{CF})$ *is closed under the following operations: union, intersection with regular languages, left and right concatenation with context-free languages, substitution with ε-free context-free languages, non-erasing homomorphism. It is not closed under intersection.*

Theorem 5.14 (Prop. 2 in [81])**.** *The emptiness problem is decidable for path controlled grammars.*

Theorem 5.15 (Prop. 3 in [81])**.** *The finiteness problem is decidable for path controlled grammars.*

Theorem 5.16 (Prop. 4 in [81])**.** *One cannot algorithmically decide whether or not the language generated by a given path controlled grammar is context-free.*

Theorem 5.17 (Prop. 5 in [81])**.** *The language generated by a given path controlled grammar can be recognized in $O(n^{10})$ time.*

Part III

New Results in Restrictions Placed upon Derivation Trees

Chapter 6

Introduction

This part being a continuation of the derivation-tree-based-restrictions investigation and represents the results written or co-written by the author. It is assumed that the reader is familiar with the graph theory (see [11]) and the theory of formal languages (see [89]), including the theory of regulated rewriting (see [26]). All preliminaries needed for understanding the following definitions, results, examples, and conclusions are presented in Part I. For the conciseness, this chapter introduces the terminology and definitions that are common for all concepts presented in the rest of this work. The examples are included everywhere it is deemed appropriate.

6.1 Common Preliminaries

Since a restriction placed upon a level, a path, and a cut is, in essence, a restriction placed upon a derivation tree, we use a slightly modified but equivalent formulation of the definitions stated in [80], [81], and [82]. Consequently, aforementioned modifications allow us to investigate all derivation-tree-based restrictions using the same terminology—e.g., restriction on the levels (see [24], [29], [104], [122], and [123]), the paths (see [17], [70], [68], [80], [81], and [82]), or the cuts (see [70]). More precisely, all restrictions placed upon the derivation trees are covered by the general notion of *tree controlled grammar* that generates its language under several kinds of the restrictions.

Note that hereafter the notion *tree controlled grammar* is used in different meaning than in Part II, see the following definitions of a *tree controlled grammar* and the definitions of the languages as well as the classes that tree controlled grammars generate under various kinds of restrictions that are introduced in the following three chapters.

The following definitions are common for the rest of this part, thus they are pointed out in its beginning.

Definition 6.1 (Tree controlled grammar)**.** A *tree controlled* grammar is a pair, (G, R), where $G = (V, T, P, S)$ is a controlled grammar, and R is a control language over V.

Definition 6.2 (Set of derivation trees)**.** Let (G, R) be a tree controlled grammar where $G = (V, T, P, S)$, then ${}_{(G,R)}\triangle(x)$, $x \in V^*$, denotes the set of the derivation trees with frontier x in G.

In the research presented through this part, we do not directly deal with level-based restriction placed upon the derivation trees. However, for the sake of completeness, note the following definitions related to level-based restriction placed upon the derivation trees.

Definition 6.3 (Language of tree controlled grammar under levels control)**.** Let (G, R) be a tree controlled grammar. The *language that* (G, R) *generates under the levels control by* R is denoted by ${}_{levels}L(G, R)$ and defined by the following equivalence:

For all $x \in T^*$, $x \in {}_{levels}L(G, R)$ if and only if there is a derivation tree, $t \in {}_{G}\triangle(x)$, such that for all levels, s, of t (except the last one), $word(s) \in R$.

Definition 6.4 (Class of tree controlled languages under levels control)**.** For some language classes $\mathbf{X}$ and $\mathbf{Y}$, the class of *tree controlled languages under the levels control* is defined as

$$\begin{aligned}\textbf{levels-TC}(\mathbf{X}, \mathbf{Y}) = \{{}_{levels}L(G, R) : {}& (G, R) \text{ is a tree controlled grammar in which} \\ & G \in \mathbb{G}_{\mathbf{X}} \text{ and } R \in \mathbf{Y}\}\end{aligned}$$

Chapter 7

Cut Tree Controlled Grammars

This chapter introduces the second part of the author's journal paper [70]:

> Koutný, J., Meduna, A.
> Tree-controlled grammars with restrictions placed upon cuts and paths.
> *Kybernetika*, 48:11, 2012.

7.1 Motivation

In this chapter, we study restrictions placed on tree cuts, and in this way, we actually open a new investigation area concerning restrictions placed upon the derivation trees. Indeed, all the other related studies discussed the restrictions placed on paths or levels, not on cuts (see [17], [24], [29], [68], [80], [81], [82], [104], [122], and [123]).

As a cut of a tree is one of the basic tree property as well as a level and a path, the natural question is to study also cut-based restrictions placed upon the derivation trees. The goal of the study in this area is to compare the impact of controlling the cuts in the derivation trees on the generative power of context-free grammars and, consequently, compare the results with level-based and paths-based restrictions placed upon the derivation trees.

More specifically, we introduce the notion of a tree controlled grammar in which we restrict its derivation-tree cuts by a prescribed regular language so that for each derivation tree in the grammar there is a set X of tree cuts that cover all the tree and X is described by given regular language. Then, we consider all these grammars and prove that they characterize the class of recursively enumerable languages. Finally, we introduce a binary relation over the derivation-tree cuts in these grammars and prove that the class of languages generated by them is also identical with the class of recursively enumerable languages.

7.2 Definitions

In this section, we introduce new derivation-tree-based restrictions of tree-controlled grammars which are based on the restriction placed upon the cuts.

Definition 7.1 (Language of tree controlled grammar under cuts control)**.** Let (G, R) be a tree controlled grammar. The *language that* (G, R) *generates under the cuts control by* R is denoted by ${}_{cut}L(G, R)$ and defined by the following equivalence:

For all $x \in T^*$, $x \in {}_{cut}L(G, R)$ if and only if there is a derivation tree, $t \in {}_{G}\triangle(x)$, and a set, ${}_{x}M$, of its cuts such that

1. for each $c \in {}_xM$, word$(c) \in R$, and
2. ${}_xM$ covers the whole t.

In other words, 1. states that ${}_xM$ contains only those cuts, which are described by R and the meaning of 2. is that if n is a node of t, then there is $c \in {}_xM$ such that c contains n.

Definition 7.2 (Class of tree controlled languages under cuts control)**.** The class of *tree controlled languages under the cuts control* is defined as

$$\textbf{cut-TC}(\textbf{CF}, \textbf{REG}) = \{{}_{cut}L(G,R) : (G,R) \text{ is a tree controlled grammar in which } G \text{ is a context-free grammar and } R \in \textbf{REG}\}$$

and the *class of tree controlled languages with ε-free controlled grammar under cuts control* is defined as

$$\textbf{cut-TC}_\varepsilon(\textbf{CF}_\varepsilon, \textbf{REG}) = \{{}_{cut}L(G,R) : (G,R) \text{ is a tree controlled grammar in which } G \text{ is an } \varepsilon\text{-free context-free grammar and } R \in \textbf{REG}\}.$$

Definition 7.3 (Ordering relation on the cuts)**.** Let $\preceq$ be a binary relation on a sequence, ${}_xM$, of the cuts such that for each two cuts, $c_1, c_2 \in {}_xM$, $c_1 \preceq c_2$ if and only if for each node, n_2, of c_2 either there is a node, n_1, of c_1 such that n_2 is a direct descendent of n_1, or $n_1 = n_2$. In other words, $n_1 \neq n_2$ implies n_2 is a direct descendent of n_1.

Definition 7.4 (Language of tree controlled grammar under ordered-cuts control)**.** Let (G,R) be a tree controlled grammar. The *language that (G,R) generates under the ordered-cuts control by R* is denoted by ${}_{ord\text{-}cut}L(G,R)$ and defined by the following equivalence:

For all $x \in T^*$, $x \in {}_{ord\text{-}cut}L(G,R)$ if and only if there is a derivation tree, $t \in {}_G\triangle(x)$, and a sequence, $c_{1_x}, c_{2_x}, \ldots, c_{n_x}$, of the cuts of t, for some $n_x \geq 1$, such that

1. for all $i = 1_x, 2_x, \ldots, n_x$, word$(c_i) \in R$,
2. $\{c_{1_x}, c_{2_x}, \ldots, c_{n_x}\}$ covers the whole t, and
3. $c_{i_x} \preceq c_{(i+1)_x}$ for all $i = 1, 2, \ldots, n-1$.

In other words, 1. states that a sequence of the cuts contains only those cuts, which are described by R, 2. says that the set defined by a sequence of the cuts covers the whole t, and the meaning of 3. is that the cuts in a sequence do not cross, although they can have some common nodes.

Definition 7.5 (Class of tree controlled languages under ordered-cuts control)**.** The class of *tree controlled languages under the ordered cuts control* is defined as

$$\textbf{ord-cut-TC}(\textbf{CF}, \textbf{REG}) = \{{}_{ord\text{-}cut}L(G,R) : (G,R) \text{ is a tree controlled grammar in which } G \text{ is a context-free grammar and } R \in \textbf{REG}\}$$

and the *class of tree controlled languages with ε-free controlled grammar under ordered cuts control* is defined as

$$\textbf{ord-cut-TC}_\varepsilon(\textbf{CF}_\varepsilon, \textbf{REG}) = \{{}_{ord\text{-}cut}L(G,R) : (G,R) \text{ is a tree controlled grammar in which } G \text{ is an } \varepsilon\text{-free context-free grammar and } R \in \textbf{REG}\}.$$

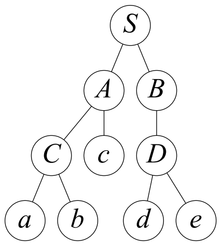

Figure 7.1: An illustration of cut-based restrictions.

To illustrate Def. 7.1 and Def. 7.4 above, suppose that in a tree controlled grammar, (G, R), there is a derivation tree given in Fig. 7.1, where *abcde* is a word composed of terminal symbols.

- In Def. 7.1, to have *abcde* in ${}_{cut}L(G,R)$, for example the set ${}_xM = \{S, AD, CcB, abcB, Ade\}$ with $\mathrm{word}(s) \in R$, for all $s \in {}_xM$, is correct. Note that, however, ${}_xM$ is not correct in terms of Def. 7.4 since the cuts cross each other in ${}_xM$.
- In Def. 7.4, to have *abcde* in ${}_{ord\text{-}cut}L(G,R)$, for example the sequence ${}_xM = S$, AB, CcB, $abcD$, $abcde$ with $\mathrm{word}(s) \in R$, for each item s of ${}_xM$, is correct, since the cuts do not cross.

7.3 Results

This section presents the results achieved in the investigation area defined in the previous section.

7.3.1 Generative Power

Theorem 7.1. $\mathbf{RE} = \mathbf{ord\text{-}cut\text{-}TC}(\mathbf{CF}, \mathbf{REG})$

Proof. Let L be a recursively enumerable language. Without any loss of generality, we assume that L is generated by an unrestricted grammar, $G = (V, T, P, S)$, in Pentonnen normal form (see Sec. 3.27). Let (G', R) be a tree controlled grammar that generates ${}_{ord\text{-}cut}L(G', R) \in \mathbf{ord\text{-}cut\text{-}TC}(\mathbf{CF}, \mathbf{REG})$, where $G' = (V', T, P', S)$, $V' = V \cup Q$, $Q = \{\langle A, B, C\rangle : AB \to AC \in P\}$, and P' is defined in the following way:

1. for $A \to x \in P$, $A \in V - T$, $x \in \{\varepsilon\} \cup T \cup (V - T)^2$, add $A \to x$ to P',
2. for $AB \to AC \in P$, $A, B, C \in (V - T)$, add the set of two productions $\{B \to \langle A, B, C\rangle$, $\langle A, B, C\rangle \to C\}$ to P'.

Without any loss of generality, we assume that $Q \cap V = \emptyset$. The regular language R is defined as follows:

$$R = V^* \cup \{V^* A\langle A, B, C\rangle V^* : AB \to AC \in P, A \in V - T, \langle A, B, C\rangle \in Q\}.$$

We define the function h from $(V')^*$ into V^* by:

for all $C \in V$, $h(C) = C$,
for all $\langle A, B, C\rangle \in Q$, $h(\langle A, B, C\rangle) = C$.

To show that $L(G) = L(G', R)$, we first prove the next claim.

Claim. $S \Rightarrow^m w$, $w \in V^*$, in G, if and only if $S \Rightarrow^n v$, $v \in (V')^*$, in (G', R), where $w = h(v)$, $v \in R$, for $m, n \geq 0$.

Only-If Part: That is, if $S \Rightarrow^m w$, $w \in V^*$, in G, then $S \Rightarrow^* v$, $v \in (V')^*$, in (G', R), where $w = h(v)$, $v \in R$, for $m \geq 0$. This is established by induction on $m \geq 0$.

Basis: Let $m = 0$. The only w is S since $S \Rightarrow^0 S$ in G. Clearly, $S \Rightarrow^0 S$ in (G', R) with $S = h(S)$, and since $S \in V^*$, $S \in R$.

Induction Hypothesis: Let us suppose that the only-if part holds for all derivations of length m or less, for some $m \geq 0$.

Induction Step: Consider a derivation $S \Rightarrow^{m+1} x$ in G, $x \in V^*$. Since $m + 1 \geq 1$, there is some $y \in V^+$ and $p \in P$ such that $S \Rightarrow^m y \Rightarrow x\,[p]$ in G, and by the induction hypothesis, $S \Rightarrow^* y'$, $y' \in (V')^*$, in (G', R) with $h(y') = y$ and $y' \in R$. Next, as far as p is concerned, we distinguish two cases:

1. p is of the form $AB \to AC$, $A, B, C \in V - T$,
2. p is of the form $A \to \alpha$, $A \in V - T$, $\alpha \in \{\varepsilon\} \cup T \cup (V - T)^2$.

Let us discuss 1. through 2. in detail.

1. Let p be of the form $AB \to AC$, $A, B, C \in V - T$. Then, $y' = y_1 A B y_2 \in R$, $y_1, y_2 \in (V')^*$, and $B \to \langle A, B, C \rangle \in P'$ is applied in (G', R). Thus, we obtain $x' = y_1 A \langle A, B, C \rangle y_2$, with $h(x') = x$ and since $x' \in V^* A \langle A, B, C \rangle V^*$, $x' \in R$. For each $\langle A, B, C \rangle \in Q$, there is $\langle A, B, C \rangle \to C \in P'$ with $h(\langle A, B, C \rangle) = h(C) = C$. Thus, $x' \Rightarrow z'$ with $h(z') = h(x') = x$, and since $z' \in V^*$, $z' \in R$.
2. Let p be of the form $A \to \alpha$, $A \in V - T$, $\alpha \in \{\varepsilon\} \cup T \cup (V - T)^2$. Then, $y' = y_1 A y_2 \in R$, $y_1, y_2 \in (V')^*$, and $A \to \alpha \in P'$ is applied in (G', R). Thus, we obtain $x' = y_1 \alpha y_2$ with $h(x') = x$, and since $x' \in V^*$, $x' \in R$.

Observe that 1. through 2. cover all possible forms of p so that the only-if part holds true.

If Part: That is, if $S \Rightarrow^n v$, $v \in (V')^*$, in (G', R), then $S \Rightarrow^* w$, $w \in V^*$, in G where $w = h(v)$, $v \in R$, for $n \geq 0$. This is established by induction on $n \geq 0$.

Basis: For $n = 0$, the only v is S since $S \Rightarrow^0 S$ in (G', R), with $h(S) = S$ and since $S \in V^*$, $S \in R$. Clearly, $S \Rightarrow^0 S$ in G.

Induction Hypothesis: Let us suppose that the if part holds for all derivations of length n or less, for some $n \geq 0$.

Induction Step: Consider a derivation of the form $S \Rightarrow^{n+1} x'$ in (G', R), where $x' \in (V')^*$. Since $n + 1 \geq 1$, there is some $y' \in V^+$ such that $S \Rightarrow^n y' \Rightarrow x'\,[p]$ in (G', R) and $y' \in R$, and by the induction hypothesis, $S \Rightarrow^* y$ in G with $h(y') = y$. Next, as far as p is concerned, we distinguish three cases:

1. p is of the form $B \to \langle A, B, C \rangle$, $B \in V'$, $\langle A, B, C \rangle \in Q$,
2. p is of the form $\langle A, B, C \rangle \to C$, $\langle A, B, C \rangle \in Q$, $C \in V'$,
3. p is of the form $A \to \alpha$, $A \in V'$, $\alpha \in \{\varepsilon\} \cup T \cup (V - T)^2$.

Let us discuss 1. through 3. in detail.

1. Let p be of the form $B \to \langle A, B, C\rangle$, $B \in V'$, $\langle A, B, C\rangle \in Q$. Then, $y = y_1By_2$ and since $x' \in R$, y_1 is of the form $y_1 = z_1A$, for some $z_1 \in V^*$. Thus, $p : AB \to AC \in P$ is applied in G and $x = y_1Cy_2$ with $h(x') = x$, $y_1 = z_1A$.

2. Let p be of the form $\langle A, B, C\rangle \to C$, $\langle A, B, C\rangle \in Q$, $C \in V'$. Then, $y = y_1Cy_2 = x$ with $h(x') = x$.

3. Let p be of the form $A \to \alpha$, $A \in V'$, $\alpha \in \{\varepsilon\} \cup T \cup (V - T)^2$. Then, $y = y_1Ay_2$ and $p : A \to \alpha$ is applied in G. Thus, $x = y_1\alpha y_2$ with $h(x') = x$.

Observe that 1. through 3. cover all possible forms of p so that the if part holds true.

The proof of the inclusion $\mathbf{RE} \subseteq \textbf{ord-cut-TC}(\mathbf{CF}, \mathbf{REG})$ can be easily obtained from the claim above. From the definition of a cut, the following properties straightforwardly follow:

1. Every sentential form is a special case of a cut. Therefore, let ${}_wM$ be the sequence of all sentential forms corresponding to the derivation tree of any $w \in L(G)$.

2. ${}_wM$ covers each node of the derivation tree of w in G at least once. Thus, ${}_wM$ covers the derivation tree of any $w \in L(G)$.

3. Considering the order of sentential forms of $w \in L(G)$ in the derivation $S \Rightarrow^* w$ in G, ${}_xM$ satisfies condition 3. stated in the definition of ${}_{\textit{ord-cut}}L(G, R)$.

Thus, $S \Rightarrow^* w$ in G if and only if $S \Rightarrow^* w$ in (G', R), $w \in T^*$. Therefore, $L(G) = {}_{\textit{ord-cut}}L(G', R)$ and consequently $\mathbf{RE} \subseteq \textbf{ord-cut-TC}(\mathbf{CF}, \mathbf{REG})$.

Clearly, $\textbf{ord-cut-TC}(\mathbf{CF}, \mathbf{REG}) \subseteq \mathbf{RE}$ and, thus, $\mathbf{RE} = \textbf{ord-cut-TC}(\mathbf{CF}, \mathbf{REG})$. □

Theorem 7.2. $\mathbf{RE} = \textbf{cut-TC}(\mathbf{CF}, \mathbf{REG})$

Proof. Clearly, $\textbf{cut-TC}(\mathbf{CF}, \mathbf{REG}) \subseteq \mathbf{RE}$. Obviously, by the definitions Def. 7.1 and Def. 7.4 in the previous section, $\textbf{ord-cut-TC}(\mathbf{CF}, \mathbf{REG}) \subseteq \textbf{cut-TC}(\mathbf{CF}, \mathbf{REG})$. Thus $\mathbf{RE} \subseteq \textbf{cut-TC}(\mathbf{CF}, \mathbf{REG})$ follows from $\mathbf{RE} \subseteq \textbf{ord-cut-TC}(\mathbf{CF}, \mathbf{REG})$ (see Th. 7.1). Therefore, $\mathbf{RE} = \textbf{cut-TC}(\mathbf{CF}, \mathbf{REG})$. □

Corollary 7.3. $\textbf{ord-cut-TC}(\mathbf{CF}, \mathbf{REG}) = \textbf{cut-TC}(\mathbf{CF}, \mathbf{REG})$

Proof. It straightforwardly follows from $\mathbf{RE} = \textbf{ord-cut-TC}(\mathbf{CF}, \mathbf{REG})$ (Th. 7.1) and $\mathbf{RE} = \textbf{cut-TC}(\mathbf{CF}, \mathbf{REG})$ (Th. 7.2). □

Discussion, Notes, and Further Research Ideas

We have introduced two types of cut-based restrictions on the derivation trees of context-free grammars, and we have proved that both of them increase the generative power of context-free grammars so they characterize **RE** (see Th. 7.1 and Th. 7.2).

A crucially important open problem area consists of the determination of the generative power of these grammars without ε-productions. In other words, future investigations concerning this subject should try to place $\textbf{ord-cut-TC}_\varepsilon(\mathbf{CF}_\varepsilon, \mathbf{REG})$ as well as $\textbf{cut-TC}_\varepsilon(\mathbf{CF}_\varepsilon, \mathbf{REG})$ into the relation with some other well-known language families, such as **CS** (see Sec. 3.3).

Obviously, $\mathbf{CS} \subseteq \textbf{ord-cut-TC}_\varepsilon(\mathbf{CF}_\varepsilon, \mathbf{REG})$ can be established by analogy with demonstrating $\mathbf{RE} \subseteq \textbf{ord-cut-TC}(\mathbf{CF}, \mathbf{REG})$ in the proof of Th. 7.1, in which covering the whole derivation tree by sentential forms is considered. Indeed, the only difference between Penttonen normal form for general grammars and context-sensitive grammars (see Def. 3.27 and Def. 3.34) is that in the former, the productions of the form $A \to \varepsilon$ are allowed; while in the later, they are not. An open problem is whether or not $\textbf{ord-cut-TC}_\varepsilon(\mathbf{CF}_\varepsilon, \mathbf{REG}) \subseteq \mathbf{CS}$ holds, which would mean $\textbf{ord-cut-TC}_\varepsilon(\mathbf{CF}_\varepsilon, \mathbf{REG}) = \mathbf{CS}$.

Clearly, $\mathbf{CS} \subseteq \textbf{cut-TC}_\varepsilon(\mathbf{CF}_\varepsilon, \mathbf{REG})$ can be demonstrated similarly as establishing $\mathbf{RE} \subseteq \textbf{cut-TC}(\mathbf{CF}, \mathbf{REG})$ in the proof of Th. 7.2, in which $\textbf{ord-cut-TC}(\mathbf{CF}, \mathbf{REG}) \subseteq \textbf{cut-TC}(\mathbf{CF}, \mathbf{REG})$ is considered. Obviously, based upon a similar argument, we can demonstrate $\textbf{ord-cut-TC}_\varepsilon(\mathbf{CF}_\varepsilon, \mathbf{REG}) \subseteq \textbf{cut-TC}_\varepsilon(\mathbf{CF}_\varepsilon, \mathbf{REG})$. An open problem is whether or not $\textbf{cut-TC}_\varepsilon(\mathbf{CF}_\varepsilon, \mathbf{REG}) \subseteq \mathbf{CS}$ holds, which would imply $\textbf{cut-TC}_\varepsilon(\mathbf{CF}_\varepsilon, \mathbf{REG})$ $= \mathbf{CS}$.

Furthermore, the results stated in Th. 7.1 and Th. 7.2 can be strengthen by considering a transformation into Geffert normal form (see [43]). As opposed to Pentonnen normal form (see Def. 3.27), in Geffert normal form only a limited number of nonterminals is allowed. This way of transformation would imply the following:

Conjecture 7.4. *For any recursively enumerable language, L, there is a tree controlled grammar, (G, R), with a context-free grammar, $G = (V, T, P, S)$, that generates the language under cuts control by R such that* $\mathrm{card}(V - T) \in \{3, 4, 5\}$ *(depending on the variant of Geffert normal form used) with ${}_{cut}L(G, R) = L$.*

Similarly as in the investigation of controlling the levels of the derivation trees of context-free grammars (see Chap. 4) by regular languages, controlling the cuts leads to increasing the generative power of context-free grammars so the resulting model characterize the class of recursively enumerable languages (see Th. 7.1 and Th. 7.2). Thus, inspired by [29] and [27] (see Sec. 4.3), an important open problem area consists of the determination of the generative power of cut-based restrictions on the derivation trees of context-free grammars in which several types of subregular languages (see Sec. 3.7) are used to control the cuts. This would lead to establishing a hierarchy of language classes similar as it is stated in [29] (see Sec. 4.3). Consequently, some interesting relation between level-based and cut-based restrictions on the derivation trees would be found out.

Chapter 8

Path Tree Controlled Grammars

This chapter introduces the author's journal paper [69] that is in press in the time of finishing this doctoral thesis:

> Koutný, J., Meduna, A.
> On normal forms and erasing rules in path controlled grammars.
> *Schedae Informaticae*, in press.

Next, this chapter introduces the author's conference paper [66]:

> Koutný, J.
> On path-controlled grammars and pseudoknots.
> In *Proceedings of the 18th Conference STUDENT EEICT 2012 Volume 3*, 2012.

Finally, this chapter introduces the second part of the author's journal paper [67] that is under consideration in the time of finishing this doctoral thesis:

> Koutný, J., Křivka, Z., and Meduna, A.
> On grammars with controlled paths.
> *Acta Cybernetica*, submitted.

8.1 Motivation

The motivation to study path-based restrictions including the state of the art concerning this topic is summarized in Sec. 1.1.2. The present chapter is based on some open problems pointed out in [80], [81], and [82], therefore it represents a natural continuation of the investigation in this research area.

Erasing Productions

In the theory of regulated rewriting, the impact of ε-productions on the generative power of a rewriting model is usually studied. However, for grammars with controlled paths, the impact of ε-productions on the generative power has not been studyied yet. Typically, beyond the class of context-free grammars, ε-productions significantly affect the generative power of a studied model (see [26], [72], [88], [119], and [139]). Moreover, grammars without ε-productions are crucially important from the practical viewpoint. Indeed, in the terms of language-recognition-related studies such as syntax analysis in the theory of compilers, the underlying grammars containing no ε-productions imply much easier construction of a parser and, as a consequence, much easier implementation.

We demonstrate ε-productions do not affect the generative power of path controlled grammars. Indeed, ε-productions outside the controlled path can be removed by a well-known algorithm (see Alg. 5.1.3.2.3 in [89]) and since a path is defined as a sequence of non-terminal symbols followed by just one terminal symbol (see Def 2.18), ε-productions cannot be used along the controlled path by the definition. Thus, this result is significant for the study given in Sec. 9.3.4 that deals with a polynomial time parsing methods for the grammars with controlled paths. Indeed, a path controlled grammar, (G, R), can be easily transformed into an equivalent path controlled grammar, (G', R), in which G' contains no ε-productions.

Normal Forms

Normal forms (see Sec. 3.27, Sec. 3.34, and Sec. 3.4) for any formal model are one of the fundamental model-characterizing properties (see [10], [22], [31], [42], [50], and [124]) which is important both from the theoretical as well as practical viewpoints. From the theoretical viewpoint, normal forms are often used to facilitate some kind of proofs—typically, the proofs based on the transformation of investigated formal model to a well-known one or vice versa. From the practical viewpoint, normal forms underlie some general parsing methods used in compiler construction (see [90]). Since path controlled grammars generate several non-context-free languages used in linguistics (see Sec. 5.2), parsing methods for path controlled grammars are desirable to be established (see [17]).

Despite all the effort so far, we are not able to transform a path controlled grammar into an equivalent path controlled grammar with controlled grammar in Chomsky normal form (see Sec. 3.4). Indeed, we have concluded that we need either unit productions or ε-productions. Then, based on these assumptions, we have formulated two normal forms for path controlled grammars including corresponding transformation algorithms. Roughly speaking, compared to Chomsky normal form (see Sec. 3.4), 1^{st} normal form adds only unit productions (see Conv. for Def. 3.35), 2^{nd} normal form adds just one ε-production (see Conv. for Def. 3.35).

Relation to Pseudoknots

A pseudoknot is introduced as the turnip yellow mosaic virus (see [117]) and it is a nucleic acid secondary structure with two or more stem-loop structures such that half of one stem is inserted between the two halves of another stem. Although pseudoknots form knot-shaped three-dimensional patterns, they are not true topological knots. The biological significance of pseudoknots rely on RNA molecules that form pseudoknots (see [54]). The fundamental problem in the pseudoknot theory in relation to the formal language theory is identification of a pseudoknot—the membership problem in terms of the theoretical computer science. It is well-known that the general problem of predicting the lowest free energy structures with pseudoknots is NP-complete (see [77] and [78]; the definition of NP-completeness can be found in Sec. 10.3.3.1 in [89]).

Inspired by biology (see [117]), we just present some pseudoknots in the form of word representation (see [57] for formal definition of general pseudoknot). However, as opposed to biology where RNA is based on the finite alphabet (adenin—A, guanin—G, cytosin—C, and uracil—U), we generalize the pseudoknots over arbitrarily alphabet, Σ. The pseudoknots are defined both as stem-only form as well as the form with arbitrarily word between the stems. Then, we demonstrate some typical pseudoknots generated by path controlled grammars for which the membership problem is decidable in a polynomial time (see [81]). Further detailed information concerning the computer-science-related investigation of pseudoknots can be found in [34], [53], and [79].

Counterargument to the Proof of Generative Power

The generative power of path controlled grammars is investigated in Prop. 6 in [80]. There, it is stated that the class of the languages generated by path controlled grammars with context-free components is strictly included in the class of the languages generated by matrix grammars. The proof is given by the transformation of a path controlled grammar into an equivalent matrix grammar.

However, the construction used in the proof does not seem to be correct. On the other hand, despite all efforts to repair the construction making the proof correct, we are not able to transform a path controlled grammar to an equivalent matrix grammar. Therefore, we describe the counterargument against the correctness of this result's proof. Then, we conclude that the generative power of path controlled grammars still represents a crucially important open problem.

8.2 Definitions

In this section, we introduce a path-based restriction on tree-controlled grammars that is equivalent to the model introduced in Chap. 5. Then, we formally define the pseudoknot structure represented as a language.

Definition 8.1 (Language of tree controlled grammar under path control)**.** Let (G, R) be a tree controlled grammar. The *language that* (G, R) *generates under the path control by* R is denoted by ${}_{path}L(G, R)$ and defined by the following equivalence:

For all $x \in T^*$, $x \in {}_{path}L(G, R)$ if and only if there is a derivation tree, $t \in {}_G\triangle(x)$, such that there is a path, p, of t with $\text{word}(p) \in R$.

Definition 8.2 (Class of tree controlled languages under path control)**.** For $\mathbf{X}, \mathbf{Y} \in \{\mathbf{LIN}, \mathbf{CF}\}$, the *class of tree controlled languages under the path control* is defined as

$$\textbf{path-TC}(\mathbf{X}, \mathbf{Y}) = \{{}_{path}L(G, R) : (G, R) \text{ is a tree controlled grammar in which } G \in \mathbb{G}_{\mathbf{X}} \text{ and } R \in \mathbf{Y}\}$$

and the *class of tree controlled languages with* ε*-free controlled grammar under path control* is defined as

$$\textbf{path-TC}_\varepsilon(\mathbf{CF}_\varepsilon, \mathbf{Y}) = \{{}_{path}L(G, R) : (G, R) \text{ is a tree controlled grammar in which } G \text{ is an } \varepsilon\text{-free context-free grammar and } R \in \mathbf{Y}\}.$$

Definition 8.3 (1st normal form of a tree controlled grammar that generates the language under path control)**.** Let (G, R) be a tree controlled grammar that generates the language under path control by R, where $G = (V, T, P, S)$. (G, R) is in 1st *normal form* if every production, $r : A \to x \in P$, is of the form $A \in V - T$ and $x \in T \cup (V - T) \cup (V - T)^2$.

Definition 8.4 (2nd normal form of a tree controlled grammar that generates the language under path control)**.** Let (G, R) be a tree controlled grammar that generates the language under path control by R, where $G = (V, T, P, S)$. (G, R) is in 2^{nd} *normal form* if every production, $r : A \to x \in P$, is of the form $A \in V - T$ and $x \in T \cup ((V \cup \{E\}) - T)^2$ where $E \cap V = \emptyset$ and $E \to \varepsilon \in P$. The alphabet of G should now include E, with $E \notin V$.

Definition 8.5 (Pseudoknot)**.** Let Σ be an alphabet. The following languages over Σ (see Fig. 8.1) are pseudoknots:

1. $\{xyx^Ry^R :\ x, y \in \Sigma^*\}$,
   $\{u_1xu_2yu_3x^Ru_4y^Ru_5 :\ x, y, u_i \in \Sigma^*, 1 \le i \le 5\}$,
2. $\{xyx^Rzz^Ry^R :\ x, y, z \in \Sigma^*\}$,
   $\{u_1xu_2yu_3x^Ru_4zu_5z^Ru_6y^Ru_7 :\ x, y, z, u_i \in \Sigma^*, 1 \le i \le 7\}$,
3. $\{xyx^Rzy^Rz^R :\ x, y, z \in \Sigma^*\}$,
   $\{u_1xu_2yu_3x^Ru_4zu_5y^Ru_6z^Ru_7 :\ x, y, z, u_i \in \Sigma^*, 1 \le i \le 7\}$,
4. $\{xyzx^Ry^Rz^R :\ x, y, z \in \Sigma^*\}$,
   $\{u_1xu_2yu_3zu_4x^Ru_5y^Ru_6z^Ru_7 :\ x, y, z, u_i \in \Sigma^*, 1 \le i \le 7\}$.

Note that presented pseudoknots form obviously non-context-free languages.

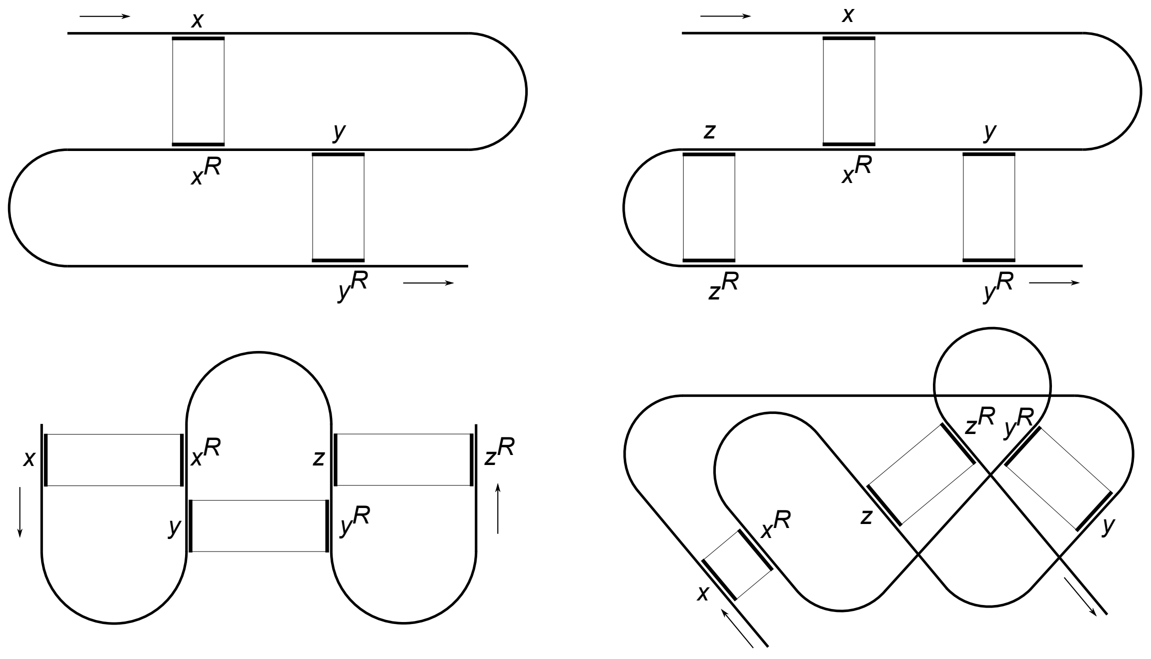

Figure 8.1: Pseudoknot examples, (top-left) $\{xyx^Ry^R :\ x, y \in \Sigma^*\}$, (top-right) $\{xyx^Rzz^Ry^R :\ x, y, z \in \Sigma^*\}$, (bottom-left) $\{xyx^Rzy^Rz^R :\ x, y, z \in \Sigma^*\}$, (bottom-right) $\{xyzx^Ry^Rz^R :\ x, y, z \in \Sigma^*\}$.

8.3 Results

This section presents the results achieved in the investigation area defined in the previous section.

8.3.1 Erasing Productions

Lemma 8.1. *For a tree controlled grammar (G, R) there is a tree controlled grammar, (G', R), such that ${}_{path}L(G, R) = {}_{path}L(G', R)$ and G' is ε-free.*

Proof. Let (G, R) where $G = (V, T, P, S)$ be a tree controlled grammar that generates ${}_{path}L(G, R)$. Without any loss of generality, assume G contains only usable productions. Basically, from all correct derivation trees for any $z \in L(G)$, we select just those of them containing a path p with $word(p) \in R$.

Consider $t \in {}_{(G,R)}\triangle(z)$, for any $z \in {}_{path}L(G, R)$. Clearly, there is a path p of t such that $word(p) = A_1 \dots A_\ell a$ with $A_1, \dots, A_\ell \in V - T$, for $\ell \geq 1$, $A_1 = S$, and $a \in T$. Consider the productions $A_i \to x_i A_{i+1} y_i$, for $1 \leq i \leq \ell - 1$, used when passing from A_i to A_{i+1} on p and, corresponding to p, the production $A_\ell \to x_\ell a\ y_\ell$ used in the last step of the derivation in G. Since $word(p) \in \{S\}(V - T)^*T$, no $A_i \to x_i A_{i+1} y_i$ is ε-production, for $1 \leq i \leq \ell - 1$.

Consider that any $x_i y_i$, $1 \leq i \leq \ell$, contains $B \in V - T$ that does not belong to p. Consider a subword z' of z that is derived from B. Since G is context-free, z' can be generated from B without using ε-productions (see the well-known Alg. 5.1.3.2.3 in [89] used for transformation of a context-free grammar to an equivalent ε-free context-free grammar).

Therefore, transformation G into G' where G' is ε-free by aforementioned algorithm cannot affect the language describing a controlled path. Thus, such a transformation cannot restrict or extend ${}_{path}L(G, R)$ properly and therefore ${}_{path}L(G, R) = {}_{path}L(G', R)$ holds. $\square$

Corollary 8.2. *For* $\mathbf{Y} \in \{\mathbf{LIN}, \mathbf{CF}\}$, $\mathbf{path\text{-}TC}(\mathbf{CF}, \mathbf{Y}) = \mathbf{path\text{-}TC}_\varepsilon(\mathbf{CF}_\varepsilon, \mathbf{Y})$.

Discussion, Notes, and Further Research Ideas

We have considered the impact of ε-productions in path controlled grammars to the generative power. As opposed to level-controlled grammars (see Chap. 4), ε-productions in path-controlled grammars do not restrict nor extend the generative power (see Lem. 8.1 and Col. 8.2).

On the other hand, when controlling the paths, the control language must be at least linear to extend the generative power (see Chap. 5) beyond context-free languages whereas for controlling levels (see Chap. 4) or cuts (see Chap. 7), a regular language is enough. As a result, we have stated that ε-productions can be removed from a path controlled grammar without affecting its language. Note that by introducing path-based restriction, the independence of context-free grammars on ε-productions have not been lost.

8.3.2 Normal Forms

Theorem 8.3. *Let* $L \in \mathbf{path\text{-}TC}(\mathbf{CF}, \mathbf{CF})$. *Then, there exists a tree controlled grammar, (G, R), in 1^{st} normal form such that $L = {}_{path}L(G, R)$.*

Proof. Alg. 1 is based on well-known Alg. 5.1.4.1.1 in [89] used for transformation of a context-free grammar to an equivalent context-free grammar in Chomsky normal form. Whenever a production $A \to X_1 X_2 \dots X_n$ for $n \geq 3$ is transformed into the productions $A \to X_1\langle X_2 \dots X_n\rangle$, $\dots$, $\langle X_{n-1} X_n\rangle \to X_{n-1} X_n$, if $word(p) = uAX_i v \in R$ for a path p, for $i = 1, 2 \dots n$, rational transducer M nondeterministically replaces AX_i as a subword of $word(p)$ by a sequence $A\langle X_2 \dots X_n\rangle\ \langle X_3 \dots X_n\rangle \dots \langle X_{i-1} \dots X_n\rangle X_i$. Whenever new nonterminal a' for $a \in V - T$ is introduced, rational transducer M replaces a as a last symbol of $word(p)$ by a sequence $a'a$. Since $\mathbf{CF}$ is closed under rational transduction, $R' \in \mathbf{CF}$. $\square$

Algorithm 1: Conversion of a tree controlled grammar (G, R) to a tree controlled grammar (G', R') in 1^{st} normal form that generates the same language under path control.

```
Input: A tree controlled grammar $(G, R)$ where $G = (V, T, P, S)$ and $G$ is $\varepsilon$-free.
Output: A tree controlled grammar $(G', R')$ where $G' = (V', T, P', S)$ satisfying
        ${}_{path}L(G, R) = {}_{path}L(G', R')$ and $(G', R')$ is in $1^{st}$ normal form.
begin
    $P' := \{r :\ r : A \to x \in P, x \in T \cup (V - T) \cup (V - T)^2\}$;
    $P_{aux} := \{r :\ r : A \to x \in P, |x| \leq 2, r \notin P'\}$;
    $V' := V$;
    assume rational transducer $M(Q, V', V', \tau, s, F)$ with $RT_M(R) = R$;
    foreach $r : A \to X_1 X_2 \dots X_n \in P$ where
    $X_i \in V$, $i = 1, 2, \dots, n$ for some $n \geq 3$ do
        begin
            $P_{aux} := P_{aux} \cup \{A \to X_1 \langle X_2 \dots X_n \rangle,$
              $\langle X_2 \dots X_n \rangle \to X_2 \langle X_3 \dots X_n \rangle,$
              $\dots$
              $\langle X_{n-2} \dots X_n \rangle \to X_{n-2} \langle X_{n-1} X_n \rangle,$
              $\langle X_{n-1} X_n \rangle \to X_{n-1} X_n\}$;
            $V' := V' \cup \{\langle X_i \dots X_n \rangle :\ i = 2, \dots, n-1\}$;
            $\tau = \tau \cup \{(f, uA\langle X_2 \dots X_n \rangle \langle X_3 \dots X_n \rangle \dots \langle X_{i-1} \dots X_n \rangle X_i v) :$
              $(f, uAX_i v) \in \tau(s, uAX_i v),$
              $f \in F, u, v \in (V')^*, A, X_i \in V'$ for some $2 \leq i \leq n\}$;
        end
    end
    foreach $r : A \to x \in P_{aux}$ with $alph(x) \cap T \neq \emptyset$ do
        begin
            replace each terminal $a \in T$ with a new symbol $a' \in V'$ in $x$;
            $V' := V' \cup \{a'\}$;
            $P' := P' \cup \{a' \to a\}$;
            $\tau := \tau \cup \{(f, ua'a) :\ (f, ua) \in \tau(s, ua), f \in F, u \in (V')^*, a \in V - T\}$;
        end
    end
    $P' := P' \cup \{r : A \to x :\ p \in P_{aux}, x \in T \cup (V')^2\}$;
    produce $G' = (V', T, P', S)$;
    produce $R' = RT_M(R)$;
    produce $(G', R')$;
end
```

Theorem 8.4. *Let $L \in$ **path-TC**$(\mathbf{CF}, \mathbf{CF})$. Then, there exists a tree controlled grammar, (G, R), in 2nd normal form such that $L = {}_{path}L(G, R)$.*

Algorithm 2: Conversion of a tree controlled grammar (G, R) to a tree controlled grammar (G', R') in 2nd normal form that generates the same language under path control.

```
Input: A tree controlled grammar $(G, R)$ in 1st normal form where $G = (V, T, P, S)$.
Output: A tree controlled grammar $(G', R')$ where $G' = (V', T, P', S)$ satisfying
        ${}_{path}L(G, R) = {}_{path}L(G', R')$ and $(G', R')$ is in 2nd normal form.
begin
    $V' := V \cup \{E : E \cap V = \emptyset\}$;
    $P' := P \cup \{E \to \varepsilon\}$;
    foreach $r : A \to x \in P$ with $x \in V - T$ do
        $P' := P' \cup \{A \to xE\}$;
    end
    produce $G' = (V', T, P', S)$;
    produce $R' = R$;
    produce $(G', R')$;
end
```

Proof. Alg. 2 is a straightforward modification of Alg. 1. First, new symbol $E \in V$ and the production $E \to \varepsilon \in P'$ are created, then each production $A \to x \in P$ with $x \in V - T$ is replaced by $A \to xE \in P'$. Clearly, this transformation does not affect the language describing a controlled path and thus $R = R' \in \mathbf{CF}$. $\square$

Examples

Next, we demonstrate two examples of typically non-context-free languages that belong to **path-TC**$(\mathbf{CF}, \mathbf{CF})$ and corresponding tree controlled grammars both in general as well as 1st and 2nd normal form. The following examples demonstrate the languages capturing multiple copy up to four parts and cross-referencing of two parts.

Example 8.1. Consider the tree controlled grammar that generates ${}_{path}L(G, R)$ where

$$\begin{aligned}
&G = (\{S, B, D, a, b, c, d\}, \{a, b, c, d\}, P, S),\\
&P = \{S \to aSd, \quad S \to aBd, \quad B \to bBc, \quad B \to D, \quad D \to bc\},\\
&R = \{S^n B^n Db : n \geq 1\}.
\end{aligned}$$

Clearly, ${}_{path}L(G, R) = \{a^k b^k c^k d^k : k \geq 1\} \notin \mathbf{CF}$.

Next, let us transform (G, R) in 1st normal form by Alg. 1 that outputs (G', R') where

$$\begin{aligned}
&G' = (\{S, B, D, \langle Sd\rangle, \langle Bd\rangle, \langle Bc\rangle, a', b', c', d', a, b, c, d, e, f\}, \{a, b, c, d, e, f\}, P', S),\\
&P' = \{S \to a'\langle Sd\rangle, \quad S \to a'\langle Bd\rangle, \quad B \to b'\langle Bc\rangle, \quad B \to D, \quad D \to b'c',\\
&\qquad \langle Sd\rangle \to Sd', \quad \langle Bd\rangle \to Bd', \quad \langle Bc\rangle \to Bc',\\
&\qquad a' \to a, \quad b' \to b, \quad c' \to c, \quad d' \to d\},\\
&R' = \{(S\langle Sd\rangle)^n S\langle Bd\rangle(B\langle Bc\rangle)^n Db'b : n \geq 1\}.
\end{aligned}$$

Clearly, ${}_{path}L(G', R') = \{a^k b^k c^k d^k : k \geq 1\} \notin \mathbf{CF}$ and (G', R') is in 1st normal form.

Finally, let us transform (G, R) in 2st normal form by Alg. 2 that outputs (G'', R'') where

$$\begin{aligned} G' &= (\{S, B, D, E, \langle Sd\rangle, \langle Bd\rangle, \langle Bc\rangle, a', b', c', d', a, b, c, d, e, f\}, \{a, b, c, d, e, f\}, P', S),\\ P'' &= \{S \to a'\langle Sd\rangle,\quad S \to a'\langle Bd\rangle,\quad B \to b'\langle Bc\rangle,\quad B \to DE,\quad D \to b'c',\\ &\qquad \langle Sd\rangle \to Sd',\quad \langle Bd\rangle \to Bd',\quad \langle Bc\rangle \to Bc',\quad E \to \varepsilon,\\ &\qquad a' \to a,\qquad b' \to b,\qquad c' \to c,\qquad d' \to d\},\\ R'' &= \{(S\langle Sd\rangle)^n S\langle Bd\rangle(B\langle Bc\rangle)^n Db'b :\ n \geq 1\}. \end{aligned}$$

Clearly, ${}_{path}L(G'', R'') = \{a^k b^k c^k d^k :\ k \geq 1\} \notin \mathbf{CF}$ and (G'', R'') is in 2^{nd} normal form.

Example 8.2. Consider the tree controlled grammar that generates ${}_{path}L(G, R)$ where

$$\begin{aligned} G &= (\{S, A, B, C, D, a, b\}, \{a, b\}, P, S),\\ P &= \{S \to aS,\quad S \to aB,\quad B \to bB,\quad B \to A,\quad A \to bA,\quad A \to C,\\ &\qquad C \to Ca,\quad C \to D,\quad D \to a\},\\ R &= \{S^m B^n A^n C^m Da :\ m, n \geq 1\}. \end{aligned}$$

Clearly, ${}_{path}L(G, R) = \{a^k b^l a^k b^l :\ k, l \geq 1\} \notin \mathbf{CF}$.

Next, let us transform (G, R) in 1^{st} normal form by Alg. 1 that outputs (G', R') where

$$\begin{aligned} G' &= (\{S, A, B, C, D, a', b', a, b\}, \{a, b\}, P', S),\\ P' &= \{S \to a'S,\quad S \to a'B,\quad B \to b'B,\quad B \to A,\quad A \to b'A,\quad A \to C,\\ &\qquad C \to Ca',\quad C \to D,\quad D \to a',\quad a' \to a,\quad b' \to b\},\\ R' &= \{S^m B^n A^n C^m Da'a :\ m, n \geq 1\}. \end{aligned}$$

Clearly, ${}_{path}L(G, R) = \{a^k b^l a^k b^l :\ k, l \geq 1\} \notin \mathbf{CF}$ and (G', R') is in 1^{st} normal form.

Finally, let us transform (G, R) in 2^{st} normal form by Alg. 2 that outputs (G'', R'') where

$$\begin{aligned} G'' &= (\{S, A, B, C, D, E, a', b', a, b\}, \{a, b\}, P'', S),\\ P'' &= \{S \to a'S,\quad S \to a'B,\quad B \to b'B,\quad B \to AE,\quad A \to b'A,\quad A \to CE,\\ &\qquad C \to Ca',\quad C \to DE,\quad D \to a',\quad a' \to a,\quad b' \to b,\quad E \to \varepsilon\},\\ R'' &= \{S^m B^n A^n C^m Da'a :\ m, n \geq 1\}. \end{aligned}$$

Clearly, ${}_{path}L(G, R) = \{a^k b^l a^k b^l :\ k, l \geq 1\} \notin \mathbf{CF}$ and (G'', R'') is in 2^{nd} normal form.

Discussion, Notes, and Further Research Ideas

We have studied two normal forms for path controlled grammars (see Def. 8.3 and Def. 8.4). Both of them are based on Chomsky normal form for context-free grammars (see Sec. 3.4). Although we were not able to establish Chomsky normal form for path controlled-grammars, we have introduced the normal form (see Th. 8.3) allowing unit productions (and no ε-productions) and the normal form (see Th. 8.4) allowing just one ε-production (and no unit productions). Then we have formulated algorithms (see Alg. 1 and Alg. 2) that transform a path controlled grammar into its normal forms.

Let us point out that it is well-known that the membership problem for path controlled grammar is decidable in a polynomial time (see [81] and [82]). In further research, both of these newly established normal forms for path controlled grammars should be taken into consideration in the relation with the results of [17]. Next, it should try to modify general parsing methods that are based on Chomsky normal form such that they will be able to parse path controlled grammars in a polynomial time.

Since there is the well-known algorithm for context-free grammars that transforms any context-free grammar in Chomsky normal form into an equivalent context-free grammar in Greibach normal form (see [89]), future investigations should consider aforementioned algorithm and reformulate it so that it modifies not only controlled grammar but also its controlling language. In other words, such an algorithm should take a path controlled grammar in general or in 1st or 2nd normal form and produce an equivalent path controlled grammar in a kind of Greibach-like (see [10], [42], [50], [124]) normal form.

8.3.3 Relation to Pseudoknots

Theorem 8.5. $\{xyx^Ry^R : \ x, y \in \Sigma^* \text{ for some } \Sigma\} \in$ **path-TC(LIN, LIN)**.

Proof. Consider the tree controlled grammar (G, R) that generates ${}_{path}L(G, R)$ where

$$
\begin{aligned}
&G = (\{S, A, B, A', B', C, D, U, V, a, b, 0, 1\}, \{a, b, 0, 1\}, P, S),\\
&P = \{1: \ S \to aA|bB,\\
&\qquad 2: \ A \to aA|aB|0C0|1D1, \qquad 3: \ B \to bB|bA|0C0|1D1,\\
&\qquad 4: \ C \to 0C0|1D1|A'|B', \qquad 5: \ D \to 1D1|0C0|A'|B',\\
&\qquad 6: \ A' \to aA'|bB'|U, \qquad 7: \ B' \to bB'|aA'|V,\\
&\qquad 8: \ U \to a, \qquad 9: \ V \to b\},\\
&R = \{Suvh(u^R)z : \ u \in \{A, B\}^*, v \in \{C, D\}^*, z \in \{Ua, Vb\}\}\\
&\qquad \text{where } h \text{ is the homomorphism defined by } h(A) = A', \ h(B) = B'.
\end{aligned}
$$

Explanation: Starting from S, (G, R) by 1 generates $w = aA$ or $w = bB$. Then, (G, R) repeatedly uses 2, 3 to generate $w = xA$ or $w = xB$ where $x \in \{a, b\}^*$ with the derivation tree containing a path Su where $u \in \{A, B\}^*$. Next, (G, R) by 2, 3 generates C or D in a sentential form and thus $w = x0C0$ or $w = x1D1$ where $x \in \{a, b\}^*$ with the derivation tree containing a path SuC or SuD where $u \in \{A, B\}^*$, respectively. Then, (G, R) repeatedly uses 4, 5 to generate $w = xyCy$ or $w = xyDy^R$ where $x \in \{a, b\}^*$, $y \in \{0, 1\}^*$ with the derivation tree containing a path Suv where $u \in \{A, B\}^*$, $v \in \{C, D\}^*$. By 4, 5, (G, R) generates $w = xyA'y^R$ or $w = xyB'y^R$ where $x \in \{a, b\}^*$, $y \in \{0, 1\}^*$ with the derivation tree containing a path $SuvA'$ or $SuvB'$ where $u \in \{A, B\}^*$, $v \in \{C, D\}^*$, respectively. Then, (G, R) uses 6, 7 to generate $w = xyx'A'y^R$ or $w = xyx'B'y^R$ where $x, x' \in \{a, b\}^*$, $y \in \{0, 1\}^*$ with the derivation tree containing a path $Suvu'$ where $u \in \{A, B\}^*$, $v \in \{C, D\}^*$, $u' \in \{A', B'\}^*$, and the equivalence $u' = h(u^R)$ is ensured by the controlling language R. Next, (G, R) uses 6, 7 to generate $w = xyx'Uy^R$ or $w = xyx'Vy^R$ where $x, x' \in \{a, b\}^*$, $y \in \{0, 1\}^*$ with the derivation tree containing a path $Suvu'U$ or $Suvu'V$, respectively, where $u \in \{A, B\}^*$, $v \in \{C, D\}^*$, $u' \in \{A', B'\}^*$, and $u' = h(u^R)$. Finally, (G, R) uses 8, 9 to generate $w = xyx^Ry^R \in T^*$ with the derivation tree containing a path $Suvu'Ua$ or $Suvu'Vb$ where $u \in \{A, B\}^*$, $v \in \{C, D\}^*$, $u' \in \{A', B'\}^*$ with $u = h(u^R)$. Thus, (G, R) generates ${}_{path}L(G, R) = \{w : \ w = xyx^Ry^R, x \in \{a, b\}^*, y \in \{0, 1\}^*\}$ that forms the pseudoknot. Clearly, both G and R are linear. □

Using the same idea as in the proof of Th. 9.2, we can demonstrate the following.

Theorem 8.6. $\{xyx^Rzz^Ry^R : \ x, y, z \in \Sigma^* \text{ for some } \Sigma\} \in$ **path-TC(LIN, LIN)**.

Theorem 8.7. $\{xyx^Rzy^Rz^R : \ x, y, z \in \Sigma^* \text{ for some } \Sigma\} \in$ **path-TC(LIN, LIN)**.

Proof. Since the idea of the construction is the same, tree controlled grammars generating the pseudoknots stated in Th. 9.3 and Th. 9.4 that actually proves the theorems are omitted. However, the schemes of the derivation trees in corresponding tree controlled grammars are sketched in Fig. 8.2 where the derivation trees of linear grammars that contain a path described by linear languages are presented. □

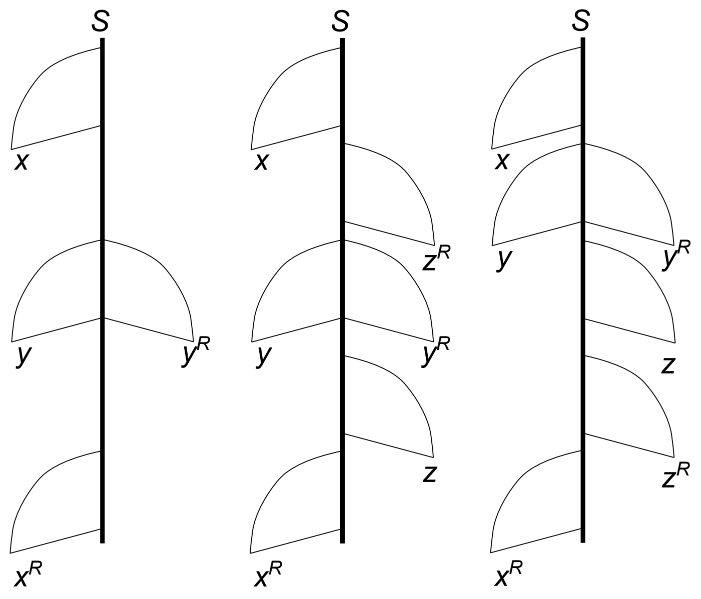

Figure 8.2: Schemes of the structure of the derivation trees of linear grammars that contain a path described by linear language, (left) $\{xyx^Ry^R : x, y \in \Sigma^* \text{ for some } \Sigma\}$, (middle) $\{xyx^Rzy^Rz^R : x, y, z \in \Sigma^* \text{ for some } \Sigma\}$, (right) $\{xyx^Rzz^Ry^R : x, y, z \in \Sigma^* \text{ for some } \Sigma\}$. Observe that the parts branched on the same level of the derivation tree (schematic view) are handled by the base linear grammar without use of the path control.

Corollary 8.8. *The pseudoknots 1 to 3 from in Def. 8.5 belong to* **path-TC**(**LIN**, **LIN**) *both in stem-only form as well as in the form with arbitrarily word between the stems.*

Discussion, Notes, and Further Research Ideas

We have demonstrated several typical pseudoknots used in biology represented by the words (see Def. 8.5). It is well-known that aforementioned pseudoknots do not belong to **CF**. Inspired by path controlled grammars introduced in [80], we have demonstrated some pseudoknots belong to **path-TC**(**LIN**, **LIN**) (see Th. 8.5, Th. 8.6, and Th. 8.7). As it clearly follows from Fig. 8.2, there are some other combinations of stem positions resulting in the language of pseudoknots-like words in **path-TC**(**LIN**, **LIN**) not mentioned in this work. However, those structures do not belong to the basic pseudoknots used in biology.

The open question is whether or not $\{xyzx^Ry^Rz^R : x, y, z \in \Sigma^*\}$ and other pseudoknot-like structures (e.g., $\{xyx^Rzy^Rwz^Rw^R : x, y, z, w \in \Sigma^*\}$) can be generated by tree controlled grammars with linear components that generate the language under path control. To answer this question, Ogdens-like lemma (see [98]) should be established and used to disprove that those languages belong to **path-TC**(**LIN**, **LIN**). If they do not, it would mean either we need stronger components (e.g., **path-TC**(**CF**, **CF**)) or even we need to control more than one path (e.g., **n-path-TC**(**CF**, **CF**) or its variants, see Chap. 9). Note that such a kind of Ogdens lemma should be significantly stronger than the lemma established in Th. 5.7 (see Prop 8 Pumping Lemma in [80]) since Ogdens lemma considers not only the subwords but also their positions (see [98]).

8.3.4 Counterargument to the Proof of Generative Power

This section being the counterargument against the result **path-TC(CF, CF)** $\subseteq$ **MAT** presented in [80]. For the completeness and better readability, using the terminology of Sec. 8.2, we present the theorem and construction part of its proof given in Prop. 6 in [80].

Theorem 8.9. (Prop. 6 in [80]) **path-TC(CF, CF)** $\subseteq$ **MAT**

Proof. Let (G, R) be a path controlled grammar with a context-free grammar, $G = (V, T, P, S)$, and a context-free grammar, $G' = (V', V, S', P')$, such that $L(G') = R$. Let $N = V - T$ and $N' = V' - V$. Without any loss of the generality, we may assume that $L(G') \subseteq \{S\}N^*T$. We define the matrix grammar $G' = \{N'' \cup T'', T'', S'', M\}$, with

$$\begin{aligned}
N'' &= N \cup N' \cup \{\overline{X}, \widehat{X} :\ X \in N \cup T\} \cup \{S''\},\\
T'' &= T \cup \{[A, B] :\ A \in N, B \in N \cup T\},\\
M &= \{(S'' \to \overline{S}S')\} \cup \{(r) :\ r \in P\} \cup \{(X \to h(x) :\ X \to x \in P'\} \cup\\
&\quad \{(\overline{A} \to u\overline{X}v, \widehat{(A)} \to [A, X]) :\ A \to uXv \in P, X \in N \cup T\} \cup\\
&\quad \{(\overline{(}a) \to a, \widehat{\to}a) :\ a \in T\}.
\end{aligned}$$

where h is the homomorphism from $(N' \cup N \cup T)^*$ into $(N' \cup \{\widehat{X} :\ X \in N \cup T\})^*$ defined by $h(X) = X$, if $X \in N'$, and $h(X) = \widehat{X}$, if $X \in N \cup T$.

After the construction part of the proof, the explanation of the idea follows. The idea tries to demonstrate that the derivation in G'' is finished only after removing all symbols with a hat from a sentence form (see Proof of Prop. 6 in [80] for the details). When the derivation is finished, the authors use a gsm mapping that reads and leaves unchanged the beginning part of its input word until the first symbol of the form $[A, B]$, where $A \in N, B \in N$, is reached. Then, the gsm mapping removes the rest of the input word, checking at the same time whether or not for each consecutive symbols $[A, B][C, D]$, where $A, C \in N, B, D \in N$, on the input tape, $B = C$; and whether or not the last two symbols of the input word are $[A, a]a$, where $A \in N, a \in T$. If both conditions are satisfied, the resulting word on the input contains only terminal symbols and since **MAT** is closed under gsm mappings, it belongs to $L(G'')$. $\square$

However, let us describe the argument against the correctness of aforementioned proof.

Example 8.3. Consider tree controlled grammar (G, R) where $G = (V, T, P, A)$ is context-free grammar with

$$\begin{aligned}
V &= \{A, B, D, X, a, b, d\},\\
T &= \{a, b, d\},\\
P &= \{A \to bB|dD|X, B \to aA, D \to aA, X \to aX|a\},
\end{aligned}$$

and $R = \{ABADAX^+a\}$. Clearly, $L(G) = \{(ba + da)^+a^+\}$ and ${}_{path}L(G, R) = \{bada^+\}$.

Without any loss of generality, consider $G_R = (V', T', P', S')$ with

$$\begin{aligned}
V' &= \{S', Y', A, B, D, X, a, b, d\},\\
T' &= \{A, B, D, X, a, b, d\},\\
P' &= \{S' \to ABADAY'a, Y' \to XY'|X\}
\end{aligned}$$

such that $L(G_R) = R$. Apparently, both components of (G, R) are context-free (i.e., G is context-free grammar and R is context-free language). Thus, as Prop. 6 in [80] states,

(G, R) can be converted to an equivalent matrix grammar G'' specified by the construction given in the proof of aforementioned proposition.

Next, we describe the counterexample proving that $L(G'') \neq {}_{path}L(G, R)$ making the proof of Prop. 6 in [80] incorrect. Let us consider $G'' = (V'' \cup T'', T'', P'', A'')$ constructed for (G, R) as in aforementioned proposition, that is

$$\begin{aligned}
V'' &= \{A, B, D, X, \overline{A}, \overline{B}, \overline{D}, \overline{X}, \overline{a}, \overline{b}, \overline{d}, \hat{A}, \hat{B}, \hat{D}, \hat{X}, \hat{a}, \hat{b}, \hat{d}, S'', S', Y'\},\\
T'' &= \{a, b, d\} \cup \{[A, A], [A, B], \dots [A, a], [B, A], [B, B], \dots [B, d], [C, A], \dots\},\\
M &= \{(A'' \rightarrow \overline{A}S')\} \cup\\
&\quad \{(A \rightarrow bB), (A \rightarrow dD), (A \rightarrow X), (B \rightarrow aA), (D \rightarrow aA), (X \rightarrow aX), (X \rightarrow a)\} \cup\\
&\quad \{(S' \rightarrow \hat{A}\hat{B}\hat{A}\hat{D}\hat{A}Y'\hat{a}), (Y' \rightarrow \hat{X}Y'), (Y' \rightarrow \hat{X})\} \cup\\
&\quad \{(\overline{A} \rightarrow \overline{b}B, \hat{A} \rightarrow [A, b]),\quad (\overline{A} \rightarrow b\overline{B}, \hat{A} \rightarrow [A, B]),\quad (\overline{A} \rightarrow \overline{d}D, \hat{A} \rightarrow [A, d]),\\
&\quad (\overline{A} \rightarrow d\hat{B}, \hat{A} \rightarrow [A, D]),\quad (\overline{A} \rightarrow \overline{X}, \hat{A} \rightarrow [A, X]),\quad (\overline{B} \rightarrow \overline{a}A, \hat{B} \rightarrow [B, a]),\\
&\quad (\overline{B} \rightarrow a\overline{A}, \hat{B} \rightarrow [B, A]),\quad (\overline{D} \rightarrow \overline{a}A, \hat{D} \rightarrow [D, a]),\quad (\overline{D} \rightarrow a\overline{A}, \hat{D} \rightarrow [D, A]),\\
&\quad (\overline{X} \rightarrow \overline{a}X, \hat{X} \rightarrow [X, a]),\quad (\overline{X} \rightarrow a\overline{X}, \hat{X} \rightarrow [X, X]), (\overline{X} \rightarrow \overline{a}, \hat{X} \rightarrow [X, a])\} \cup\\
&\quad \{(\overline{a} \rightarrow a, \hat{a} \rightarrow a), (\overline{b} \rightarrow b, \hat{b} \rightarrow b), (\overline{d} \rightarrow d, \hat{d} \rightarrow d)\}
\end{aligned}$$

Let us show ${}_{path}L(G, R) \neq L(G'')$. Indeed, consider the derivation $A \Rightarrow bB \Rightarrow baA \Rightarrow badD \Rightarrow badaA \Rightarrow badaX \Rightarrow badaaX \Rightarrow badaaa \in {}_{path}L(G, R)$ and consider its derivation tree in G.

Clearly, starting from A, G must use $A \rightarrow bB \in P$ to ensure the existence of a path starting with AB. Note that the word $dabaaa \notin {}_{path}L(G, R)$ since the only way to generate $dabaaa \in L(G)$ is to use $A \rightarrow dD$ to rewrite start nonterminal A. Next steps of the derivation are obvious.

Now, consider corresponding derivation in G'' constructed above. G'' must start by using the matrix $(A'' \rightarrow \overline{A}S')$. Then, G'' by using the matrices $(S' \rightarrow \hat{A}\hat{B}\hat{A}\hat{D}\hat{A}Y'\hat{a}), (Y' \rightarrow \hat{X}Y'), (Y' \rightarrow \hat{X})$ generates $\overline{A}\hat{A}\hat{B}\hat{A}\hat{D}\hat{A}\hat{X}\hat{X}\hat{a}$ representing the path G'' needs to check. Now, G'' by using the matrices of the form $(\overline{U} \rightarrow v\overline{V}, \hat{U} \rightarrow [U, V])$ simulates a derivation step in G and rewrite *corresponding* (this formulation is used in Prop. 6 in [80]) symbol on the path. Roughly speaking, the matrices allow to simulate a derivation step in G but they cannot ensure rewriting the *corresponding* symbol on the path. Indeed, let us assume G'' needs to rewrite $\hat{A}$ but the matrices only allow to rewrite *any* $\hat{A}$ not the *corresponding* $\hat{A}$ (actually, G'' needs to rewrite leftmost $\hat{A}$ in this case and this cannot be ensured in matrix grammars at all).

Consider Fig. 8.3 that depicts allowed derivation in G'' where the path $ABADAXXa$ needs to be checked (this corresponds to the word $\hat{A}\hat{B}\hat{A}\hat{D}\hat{A}\hat{X}\hat{X}\hat{a}$ in the right-hand part of the figure); however, the simulated derivation tree (left-hand part of the figure) contains a path $\overline{ADABAXX\overline{a}}a$ that is, not allowed in ${}_{path}L(G, R)$. Clearly, G'' produces $dabaaa[A, B][B, A][A, D][D, A][A, X][X, X][X, a]a$. Then, the gsm mapping (see the proof of Prop. 9 in [80]) removes $[A, B][B, A] \dots a$ since both the conditions stated in the aforementioned proof are fulfilled and the resulting word is $dabaaa \in L(G'')$. However, $dabaaa \notin {}_{path}L(G, R)$ as it is demonstrated above.

Discussion, Notes, and Further Research Ideas

The inclusion stated by Prop. 9 in [80] still may hold. However, it cannot be proved in the presetned way. On the other hand, the consequence stated by the application of Prop. 7 in [80] (Pumping Lemma) holds—there is the matrix language $\{a^n b^n c^n d^n e^n, n \geq 0\} \notin \textbf{path-TC}(\textbf{CF}, \textbf{CF})$. Clearly, $\textbf{path-TC}(\textbf{CF}, \textbf{CF})$ surely covers at least all context-free languages (consider V^* as a control language). However, it is still unanswered if

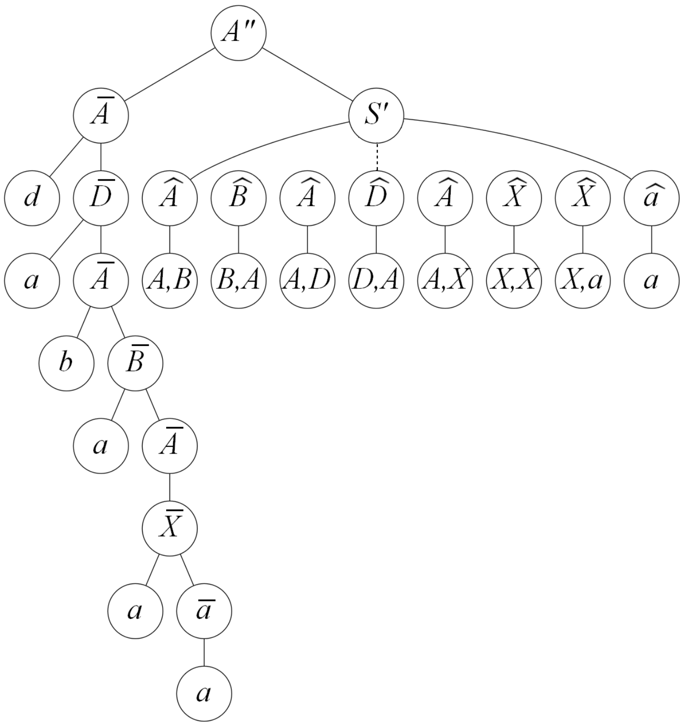

Figure 8.3: Derivation $dabaaa \in L(G'')$. However, $dabaaa \notin L(G, R)$ and thus $L(G'') \neq L(G, R)$. For the conciseness, the details of derivations $S' \Rightarrow^* \widehat{A}\widehat{B}\widehat{A}\widehat{D}\widehat{A}\widehat{X}\widehat{X}\widehat{a}$ are omitted in this figure.

path-TC$(\mathbf{CF}, \mathbf{CF})$ is strictly under or incomparable to matrix languages. Apparently, the construction used in Prop. 6 in [80] works correctly in the case when left-most matrix grammar (see [114]) is considered instead of matrix grammar. However, this is of no interest since left-most matrix grammars characterize the class of recursively enumerable languages. Any our attempt to modify aforementioned construction to work correctly ended in a failure, thus the generative power of tree controlled grammars that generate the language under path control still remains open. The only approximation so far is a conjecture obtained as a direct consequence of Th. 9.13:

Conjecture 8.10. *Let* $L \in$ **path-TC**$(\mathbf{CF}, \mathbf{CF})$. *Then, there exists* $L' = L\{\blacklozenge^q\} \in \mathbf{PSC}$ *for* $q \geq 2$.

Conjecture 8.11. *Let* $L \in$ **path-TC**$(\mathbf{CF}, \mathbf{CF})$. *Then, there exist* $L' \in \mathbf{PSC}$ *with* $L = h(L')$ *for some homomorphism* h.

Chapter 9

n-Path Tree Controlled Grammars

This chapter introduces the first part of the author's journal paper [70]:

> Koutný, J., Meduna, A.
> Tree-controlled grammars with restrictions placed upon cuts and paths.
> *Kybernetika*, 48:11, 2012.

The conference version of the paper [70] was presented at 15th Conference and Competition STUDENT EEICT 2009 [63]:

> Koutný, J.
> Regular paths in derivation trees of context-free grammars.
> In *Proceedings of the 15th Conference STUDENT EEICT 2009 Volume 4*, 2009.

Next, this chapter introduces the author's conference paper [68]:

> Koutný, J., Křivka, Z., Meduna, A.
> Pumping properties of path-restricted tree-controlled languages.
> In *7th Doctoral Workshop on Mathematical and Engineering Methods in Computer Science*, 2011.

Aforementioned conference paper [68] represents a significantly extended and revised version of the author's conference paper [64]:

> Koutný, J.
> On n-path-controlled grammars.
> In *Proceedings of the 16th Conference STUDENT EEICT 2010 Volume 5*, 2010.

Then, this chapter introduces the author's journal paper [67] that is under consideration in the time of finishing this doctoral thesis:

> Koutný, J., Křivka, Z., and Meduna, A.
> On grammars with controlled paths.
> *Acta Cybernetica*, submitted.

Finally, this chapter introduces the the author's journal paper [17]:

> Čermák, M., Koutný, J., Meduna, A.
> Parsing based on n-path tree-controlled grammars.
> *Theoretical and Applied Informatics*, 2011:16, 2012.

The conference version of the paper [17] was presented at 17th Conference and Competition STUDENT EEICT 2011 [65]:

> Koutný, J.
> Syntax analysis of tree-controlled languages.
> In *Proceedings of the 17th Conference STUDENT EEICT 2011 Volume 3*, 2011.

9.1 Motivation

In this chapter, we study restrictions placed on several paths, and in this way, we actually open a new investigation area concerning path-based restrictions placed upon the derivation trees because all the other related studies discussed the restrictions placed on just one path (see [80], [81], [82], [122], and [123]).

The derivations in context-free grammars can be seen in the notions of graph theory—by the derivation trees (see Sec. 3.4). From this viewpoint, the derivation starts in the root of a tree (which corresponds to the start symbol of a grammar), continues by sequential application of context-free productions, and ends in the leafs (which corresponds to the terminals of a grammar) without considering any context. From this rough idea, it can be easily seen that during the derivation in a context-free grammar, the paths of the corresponding derivation tree are formed. The paths of a derivation tree have several well-known properties. Namely all paths start in the common node (i.e., the root) and each of them ends in a different node (i.e., a leaf). Considering two different paths, clearly, there is a node in which both paths split. Taking this into account, we can manage some context information during the derivations in context-free grammars. Namely, the common part of all restricted paths can form the original part of each individual path and this way context information can be distributed through different branches of a derivation tree. Roughly speaking, the branches of a derivation tree communicate through their common nodes. As opposed to context-sensitive grammars; however, a derivation tree is constructed as in context-free case. Indeed, each branch of a tree can be constructed individually regardless of the other branches.

As a generalization of path controlled grammars (see Chap. 5), we introduce a model where not just one but given number of paths in the derivation trees of a context-free grammar must be described by a control language. That is, we deal with a generalization of path controlled grammars (G, R), where a word, w, generated by G belongs to the language defined by G and R only if there is a derivation tree, t, for w in G such that there exist given n paths of t described by a linear language, R, where $n \geq 1$. Such a rewriting system is referred to as n-path tree controlled grammar. Then, we demonstrate that this generalized model can generate some languages not captured in the model with one controlled path introduced in Chap. 5.

All Paths Restriction

In [80], it is established that controlling just one path of the derivation trees of context-free grammars by a linear control language increases the generative power of context free grammars (see Sec. 1.1.2). On the other hand, also in [80], it is demonstrated that controlling just one path of the derivation trees of context-free grammars by any regular control language does not increase the generative power of context-free grammars properly (see Sec. 1.1.2). Thus, in terms of controlling more than just one path in the derivation trees of context-free grammars by linear languages, the simple and natural question arises whether or not the controlling all paths in the derivation trees of context-free grammar by any regular language increases the generative power of context-free grammars.

More specifically, we restrict every root-to-leaf path in the derivation trees of context-free grammars by some control languages. We demonstrate that if these control languages are regular, the generative power of context-free grammars remains unchanged—they characterize the class of context-free languages. This result is of some interest when compared to the study given in [24], which restricts tree levels rather than paths in this way and proves that the resulting grammars characterize the class of recursively enumerable languages even when considering regular control languages. Let us also point out that our result significantly generalizes the study of [80], which only requires that there is at least one root-to-leaf path in derivation tree restricted by a regular language. Indeed, by Th. 5.2 (see Prop. 2 in [80]), if the derivation trees are restricted so they must contain at least one path in the given control regular language, then this restriction does not affect the generative power of context-free grammars. Therefore, we prove that this is true even if all paths are restricted in this way.

Pumping Properties and Closure Properties

In the formal language theory, a pumping property states that for a particular language to be a member of a language class, any sufficiently long word in the language contains a part that can be removed, or repeated any number of times, with the resulting word remaining in that language (see [9], [14], [21], [35], [47], [89], [132], [135], [137]). A pumping property for a language class is typically used to determine if a particular language does not belong to a given language class (however, it cannot be used to determine if a language belongs to a given class); and to determine if a given class is not closed under given operation over the languages (usually based on a proof by contradiction). Therefore, we state several pumping properties of some languages generated by grammars with controlled paths.

Generative Power Approximation

The generative power for one path controlled grammars is not well-known since the transformation of a grammar with one controlled path into a matrix grammar (see Sec. 8.3.4) does not satisfy the equivalence of the languages generated by both grammars. Since the generative power of any rewriting model is one of the fundamental language-characterizing properties, it is desirable to investigate also the model with restricted paths in this manner.

As opposed to matrix grammars, in path-based restrictions placed upon the derivation trees the rewriting is regulated vertically (i.e., from the root to the leafs of a tree). As it is demonstrated in (see Sec. 8.3.4), we have not been able to correct the construction based upon the transformation of a path controlled grammar into an equivalent matrix grammar. In scattered context grammars it is possible to identify the desired position to be rewritten more accurately than in matrix grammars. Therefore, we investigate the relationship between grammars with restricted paths and scattered context grammars. Moreover, it is desirable to investigate also the relationship between the language classes generated by the model with just one path restricted and the model with several path restricted.

Syntax Analysis

For a rewriting model to be usable in practice, its polynomial time parsability is crucially important property. The idea of recognition of the words generated by the grammars with controlled paths is introduced in [80] (see Chap. 5), where it is demonstrated that path controlled languages (with just one restricted path) can be recognized in a polynomial time. More detailed discussion concerning a polynomial time recognition of the languages generated by the grammars with controlled paths can be found in [81] where the membership problem is reduced to the non-emptiness problem of the intersection of two languages (see Chap. 5). As a result, [82] demonstrates that the membership problem for path controlled languages is decidable in a polynomial time. However, for the vast majority of practical applications, it is essential to find out not only whether or not given word belongs to the language of given rewriting model, but also how the model can generate it. Therefore, for a given word, it is desirable to find out which productions the model must use and in which order to generate it.

Moreover, the method presented in [81] for just one controlled path cannot be straightforwardly modified for the model with $n \geq 2$ controlled paths. Indeed, it would lead to the question if such intersection contains at least n elements that is much more difficult problem than non-emptiness. However, the problem for $n = 1$ controlled paths corresponds to the method introduced in [81]. Thus, we introduce the LL (see [4] and [89] for the definition of LL property) restriction and we present the ideas how to parse the language class the LL model generates (not only how to decide the membership problem). Essentially, we discuss the following problem: *Can we decide whether a word is recognized by a n-path tree controlled grammar in a polynomial time, for $n \geq 1$?*

9.2 Definitions

In this section, we introduce new derivation-tree-based restriction that is based on the generalization of the one path restricted model from the previous chapter to given number of paths.

Definition 9.1 (Language of tree controlled grammar under all paths control)**.** Let (G, R) be a tree controlled grammar. The *language that (G, R) generates under the all-paths control by R* is denoted by ${}_{\textit{all-paths}}L(G, R)$ and defined by the following equivalence:

For all $x \in T^*$, $x \in {}_{\textit{all-paths}}L(G, R)$ if and only if there is a derivation tree, $t \in {}_G\triangle(x)$, such that for all paths, s, of t, word$(s) \in R$.

Definition 9.2 (Class of tree controlled languages under all paths control)**.** The *class of tree controlled languages under all paths control* is defined as

$$\textbf{all-path-TC}(\textbf{CF}, \textbf{REG}) = \{{}_{\textit{all-paths}}L(G, R) : (G, R) \text{ is a tree controlled grammar in which } G \text{ is a context-free grammar and } R \in \textbf{REG}\}.$$

Definition 9.3. Let (G, R) be a tree controlled grammar. The *language of tree controlled grammar under not common n-path control* by R, $n \geq 1$, is denoted by ${}_{\textit{nc-n-path}}L(G, R)$ and defined by the following equivalence:

For all $x \in T^*$, $x \in {}_{\textit{nc-n-path}}L(G, R)$ if there exists a derivation tree, $t \in {}_G\triangle(x)$, such that there is a set, Q_t, of n paths of t such that for each path, $p \in Q_t$, it holds word$(p) \in R$.

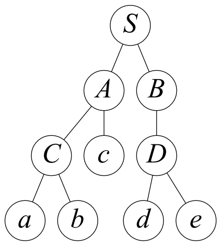

Figure 9.1: An illustration of common node of division for the paths.

Definition 9.4. For $\mathbf{X}, \mathbf{Y} \in \{\mathbf{REG}, \mathbf{LIN}, \mathbf{CF}\}$, the *class of tree controlled languages under not-common n-path control* is defined as

$$\textbf{nc-n-path-TC}(\mathbf{X}, \mathbf{Y}) = \{_{\textit{nc-n-path}}L(G,R) : (G,R) \text{ is a tree controlled grammar with } G \in \mathbb{G}_{\mathbf{X}} \text{ and } R \in \mathbf{Y}\}.$$

Definition 9.5 (Common part of two paths)**.** Let p_1, p_2 be any different two paths of a derivation tree, t. Then, p_1 and p_2 contain at least one common node (the root of t, $\mathrm{root}(t)$), and p_1 ends in a different leaf of t than p_2. Let $\mathrm{cmn}(p_1, p_2)$ denote the maximal number of consecutive common nodes of p_1 and p_2.

Definition 9.6 (Common node of division of set of paths)**.** Let Q_t be a nonempty set of some paths of a derivation tree, t. The *paths of Q_t are divided in a common node of t* if and only if for some $k \geq 1$, $\mathrm{cmn}(p_1, p_2) = k$ for every pair $(p_1, p_2) \in Q_t^2$ (see Fig. 9.1). Let all paths of Q_t be divided in a common node of t. If $Q_t = \{p\}$, then $m_{Q_t} = |\,\mathrm{word}(p)|$, otherwise $m_{Q_t} \geq 1$ denotes the maximal number of consecutive common nodes of all paths in Q_t.

Definition 9.7 (Language of tree controlled grammar under n-path control)**.** Let (G, R) be a tree controlled grammar. The *language of tree controlled grammar under n-path control* by R, $n \geq 1$, is denoted by $_{\textit{n-path}}L(G,R)$ and defined by the following equivalence:

For all $x \in T^*$, $x \in {_{\textit{n-path}}L(G,R)}$ if there exists a derivation tree, $t \in {_G\triangle(x)}$, such that there is a set, Q_t, of n paths of t that are divided in a common node of t and for each, $p \in Q_t$, it holds $\mathrm{word}(p) \in R$.

Definition 9.8 (Class of tree controlled languages under n-path control)**.** For $\mathbf{X}, \mathbf{Y} \in \{\mathbf{REG}, \mathbf{LIN}, \mathbf{CF}\}$, the *class of tree controlled languages under n-path control* is defined as

$$\textbf{n-path-TC}(\mathbf{X}, \mathbf{Y}) = \{_{\textit{n-path}}L(G,R) : (G,R) \text{ is a tree controlled grammar with } G \in \mathbb{G}_{\mathbf{X}} \text{ and } R \in \mathbf{Y}\}.$$

Conventions

Hereafter, tree controlled grammars that generate the language under the n-path control are referred to as *n-path tree controlled grammars.*

To illustrate Def. 9.1, Def. 9.6, and Def. 9.7 above, suppose that in a tree controlled grammar, (G, R), there is a derivation tree given in Fig. 9.1, where *abcde* is a word composed of terminal symbols.

- In Def. 9.1, to have *abcde* in $_{\textit{all-paths}}L(G,R)$, all of the words $SACa$, $SACb$, SAc, $SBDd$, $SBDe$ must be in R.

- In Def. 9.6, let $Q_t = \{SACa, SAc\}$, then the paths of Q_t are divided in common node and $m_{Q_t} = 2$; let $Q_t = \{SACa, SACb, SAc\}$, then the paths of Q_t are not divided in common node and m_{Q_t} is undefined.

- In Def. 9.7, to have $abcde$ in ${}_{n\text{-}path}L(G,R)$, for some $n \geq 1$, at least n of the words $SACa$, $SACb$, SAc, $SBDd$, $SBDe$ must be in R.

Note that if we consider 0 controlled paths (i.e., $n = 0$ and consequently $\text{card}(Q_t) = 0$) in the definition of ${}_{n\text{-}path}L(G,R)$, then, clearly, the generative power of such a model equals **CF**.

Since for each context-free grammar, there is a regular language that describes all paths in all its derivation trees (see Prop. 1 in [80]); and there is no regular language which increases its generative power when used to restrict the paths (see Prop. 2 in [80] and Sec. 9.3.1), if we consider tree controlled grammars (G,R) with $R \in \mathbf{REG}$, then, obviously, the generative power of such a model equals **CF** for any $n \geq 1$. Therefore, we investigate the properties of tree controlled grammar with non-regular control language. More specifically, we study tree controlled grammars that generates their languages under n-path control with linear control languages.

Definition 9.9 (Special types of n-path controlled grammars)**.** Let (G,R) be a tree controlled grammar. Consider ${}_{n\text{-}path}L(G,R)$, for $n \geq 1$. If for each word, $z \in {}_{n\text{-}path}L(G,R)$, there exist a derivation tree, $t \in {}_G\triangle(z)$, a set of its paths, Q_t, $m_{Q_t} \geq 1$, and a partition, $\text{word}(p) = uvwxy$, for each path, $p \in Q_t$, satisfying the premise of Lem. 3.2 such that it holds

$1 \leq m_{Q_t} \leq |u|$, then ${}_{n\text{-}path}L(G,R)$ is ${}_{I\text{-}n\text{-}path}L(G,R)$,
$|u| < m_{Q_t} \leq |uv|$, then ${}_{n\text{-}path}L(G,R)$ is ${}_{II\text{-}n\text{-}path}L(G,R)$,
$|uv| < m_{Q_t} \leq |uvw|$, then ${}_{n\text{-}path}L(G,R)$ is ${}_{III\text{-}n\text{-}path}L(G,R)$,
$|uvw| < m_{Q_t} \leq |uvwx|$, then ${}_{n\text{-}path}L(G,R)$ is ${}_{IV\text{-}n\text{-}path}L(G,R)$,
$|uvwx| < m_{Q_t} \leq |uvwxy|$, then ${}_{n\text{-}path}L(G,R)$ is ${}_{V\text{-}n\text{-}path}L(G,R)$.

Definition 9.10 (Classes of special types of n-path tree controlled languages)**.** For $i \in \{\mathbf{I, II, III, IV, V}\}$ and $n \geq 1$, the *class of i-n-path tree controlled languages* is defined as

$$i\textbf{-n-path-TC}(\mathbf{CF},\mathbf{LIN}) = \{{}_{i\text{-}n\text{-}path}L(G,R) : (G,R) \text{ is tree controlled grammar in which } G \text{ is a context-free grammar and } R \in \mathbf{LIN}\}.$$

Examples

We demonstrate two non-context-free languages that belong to $\mathbf{n\text{-}path\text{-}TC}(\mathbf{CF},\mathbf{CF})$, more precisely, to $\mathbf{III\text{-}n\text{-}path\text{-}TC}(\mathbf{CF},\mathbf{CF})$. First, the languages are presented for some $n \geq 1$, then we present a general case. The specific examples for higher values of n tends to be excessively long and they are left to the reader. To illustrate the definition of $\mathbf{III\text{-}n\text{-}path\text{-}TC}(\mathbf{CF},\mathbf{CF})$, consider Ex. 9.1 and Ex. 9.3 and the corresponding derivation trees in Fig. 9.2.

Example 9.1. Consider the tree controlled grammar (G,R) that generates ${}_{III\text{-}n\text{-}path}L(G,R)$ and $n = 2$, where

$$\begin{aligned}
&G = (\{S, X, Y, U, V, a, b, c, d, e, f\}, \{a, b, c, d, e, f\}, P, S),\\
&P = \{S \to aSf, \quad S \to aXYf, \quad X \to bXc, \quad Y \to dYe,\\
&\qquad X \to U, \quad U \to bc, \quad Y \to V, \quad V \to de\},\\
&R = \{S^m X^m Ub \cup S^m Y^m Vd : \ m \geq 1\},\\
&{}_{\textit{III-2-path}}L(G, R) = \{a^k b^k c^k d^k e^k f^k : \ k \geq 1\}.
\end{aligned}$$

Clearly, ${}_{\textit{III-2-path}}L(G, R) \notin \mathbf{CF}$. The left-hand part of Fig. 9.2 illustrates the derivation tree for the derivation $S \Rightarrow^* a^3b^3c^3d^3e^3f^3$ in (G, R). Clearly, there are two paths described by the words S^3X^3Ub and S^3Y^3Vd from R.

Example 9.2. Let (G, R) be a tree controlled grammar that generates ${}_{\textit{III-n-path}}L(G, R)$, $n \geq 1$, where

$$\begin{aligned}
&G = (\{S\} \cup \{A_i, B_i : \ 1 \leq i \leq n\} \cup \{a_i : \ 1 \leq i \leq 2n + 2\},\\
&\qquad \{a_i : \ 1 \leq i \leq 2n + 2\}, P, S),\\
&P = \{S \to a_1 S a_{2n+2}, \qquad S \to a_1 A_1 A_2 \dots A_n a_{2n+2}\} \cup\\
&\qquad \{A_{i+1} \to a_{2i+2} A_{i+1} a_{2i+3}, \quad A_{i+1} \to B_{i+1},\\
&\qquad B_{i+1} \to a_{2i+2} a_{2i+3} : \ 0 \leq i \leq n - 1\},\\
&R = \bigcup_{i=1}^{n} \{S^k A_i^k B_i a_{2i} : \ k \geq 1\}.
\end{aligned}$$

Clearly, $R \in \mathbf{LIN}$. Consider a derivation in (G, R):

$$\begin{aligned}
S &\Rightarrow^k a_1^k S a_{2n+2}^k\\
&\Rightarrow a_1^k a_1 A_1 A_2 \dots A_n a_{2n+2} a_{2n+2}^k\\
&\Rightarrow^{n \cdot k} a_1^{k+1} a_2^k B_1 a_3^k \dots a_{2n}^k B_n a_{2n+1}^k a_{2n+2}^{k+1}\\
&\Rightarrow^n a_1^{k+1} a_2^{k+1} a_3^{k+1} \dots a_{2n}^{k+1} a_{2n+1}^{k+1} a_{2n+2}^{k+1}
\end{aligned}$$

Clearly, n paths are described by R and in this way (G, R) generates ${}_{\textit{III-n-path}}L(G, R) = \{a_1^k a_2^k \dots a_{2n+2}^k : \ k \geq 1\} \notin \mathbf{CF}$.

Example 9.3. Consider the tree controlled grammar (G, R) with ${}_{\textit{III-n-path}}L(G, R)$, for $n = 3$, where

$$\begin{aligned}
&G = (\{A, B, C, D, E, F, G, H, I, a, b, c, d\}, \{a, b, c, d\}, P, A),\\
&P = \{A \to aA, \quad A \to aB, \quad B \to Bb, \quad B \to C,\\
&\qquad C \to cC, \quad C \to D, \quad D \to Dd, \quad D \to HHH,\\
&\qquad E \to Ea, \quad E \to I, \quad F \to bF, \quad F \to E,\\
&\qquad G \to Gc, \quad G \to F, \quad H \to dH, \quad H \to G, \quad I \to a\},\\
&R = \{A^r B^s C^t D^u H^u G^t F^s E^r Ia : \ r, s, t, u \geq 0\},\\
&{}_{\textit{III-3-path}}L(G, R) = \{(a^r c^t d^u b^s)^4 : \ r > 0, s, t, u \geq 0\}.
\end{aligned}$$

Clearly, ${}_{\textit{III-3-path}}L(G, R) \notin \mathbf{CF}$. The right-hand part of Fig. 9.2 illustrates the derivation tree for the derivation $A \Rightarrow^* (a^2cdb^2)^4$ in (G, R). Clearly, there are three paths described by $A^2B^3C^2D^2H^2G^2F^3E^2Ia$ from R.

Example 9.4. Let $m \geq 0$ with even m. Let (G, R) be a tree controlled grammar that generates ${}_{\textit{III-n-path}}L(G, R)$, $n \geq 1$, where

$$\begin{aligned}
&G = (\{A_j, B_j, a_j : \ 1 \leq j \leq m\} \cup \{C\}, \{a_j : \ 1 \leq j \leq m\}, P, A_1),\\
&P = \{A_1 \to a_1 A_1, \quad A_1 \to a_1 A_2, \quad B_1 \to B_1 a_1, \quad B_1 \to C, \quad C \to a_1\} \cup\\
&\qquad \{A_m \to A_m a_m, A_m \to \{B_m\}^n\} \cup\\
&\qquad \{A_i \to A_i a_i, \quad A_i \to A_{i+1} : \ 2 \leq i \leq m - 1 \text{ with even } i\} \cup
\end{aligned}$$

$$\begin{aligned}
&\{A_i \rightarrow a_iA_i, \quad A_i \rightarrow A_{i+1} : \ 3 \leq i \leq m-1 \text{ with odd } i\} \cup \\
&\{B_i \rightarrow a_iB_i, \quad B_i \rightarrow B_{i-1} : \ 2 \leq i \leq m \text{ with even } i\} \cup \\
&\{B_i \rightarrow B_ia_i, \quad B_i \rightarrow B_{i-1} : \ 3 \leq i \leq m \text{ with odd } i\}, \\
R = &\{A_1^{k_1}A_2^{k_2}\dots A_m^{k_m}B_m^{k_m}B_{m-1}^{k_{m-1}}\dots B_2^{k_2}B_1^{k_1}Ca_1 : \ k_i \geq 0, 1 \leq i \leq m\}.
\end{aligned}$$

Clearly, $R \in \mathbf{LIN}$. Consider a derivation in (G, R), for some even $m \geq 0$:

$$\begin{aligned}
A_1 &\Rightarrow^{k_1} a_1^{k_1}A_1 \Rightarrow a_1^{k_1+1}A_2 \Rightarrow^{k_2} a_1^{k_1+1}A_2a_2^{k_2} \Rightarrow a_1^{k_1+1}A_3a_2^{k_2} \\
&\Rightarrow^* a_1^{k_1+1}a_3^{k_3}a_5^{k_5}\dots a_{m-1}^{k_{m-1}}A_ma_m^{k_m}\dots a_6^{k_6}a_4^{k_4}a_2^{k_2} \\
&\Rightarrow a_1^{k_1+1}a_3^{k_3}a_5^{k_5}\dots a_{m-1}^{k_{m-1}}(B_m)^na_m^{k_m}\dots a_6^{k_6}a_4^{k_4}a_2^{k_2} \\
&\Rightarrow^{n\cdot k_m} a_1^{k_1+1}a_3^{k_3}a_5^{k_5}\dots a_{m-1}^{k_{m-1}}(a_m^{k_m}B_m)^na_m^{k_m}\dots a_6^{k_6}a_4^{k_4}a_2^{k_2} \\
&\Rightarrow^n a_1^{k_1+1}a_3^{k_3}a_5^{k_5}\dots a_{m-1}^{k_{m-1}}(a_m^{k_m}B_{m-1})^na_m^{k_m}\dots a_6^{k_6}a_4^{k_4}a_2^{k_2} \\
&\Rightarrow^{n\cdot k_{m-1}} a_1^{k_1+1}a_3^{k_3}a_5^{k_5}\dots a_{m-1}^{k_{m-1}}(a_m^{k_m}B_{m-1}a_{m-1}^{k_{m-1}})^na_m^{k_m}\dots a_6^{k_6}a_4^{k_4}a_2^{k_2} \\
&\Rightarrow^* a_1^{k_1+1}\dots a_{m-1}^{k_{m-1}}(a_m^{k_m}a_{m-2}^{k_{m-2}}\dots a_2^{k_2}B_1a_1^{k_1}\dots a_{m-3}^{k_{m-3}}a_{m-1}^{k_{m-1}})^na_m^{k_m}\dots a_2^{k_2} \\
&\Rightarrow^n a_1^{k_1+1}\dots a_{m-1}^{k_{m-1}}(a_m^{k_m}a_{m-2}^{k_{m-2}}\dots a_2^{k_2}Ca_1^{k_1}\dots a_{m-3}^{k_{m-3}}a_{m-1}^{k_{m-1}})^na_m^{k_m}\dots a_2^{k_2} \\
&\Rightarrow^n a_1^{k_1+1}\dots a_{m-1}^{k_{m-1}}(a_m^{k_m}a_{m-2}^{k_{m-2}}\dots a_2^{k_2}a_1^{k+1}\dots a_{m-3}^{k_{m-3}}a_{m-1}^{k_{m-1}})^na_m^{k_m}\dots a_2^{k_2}
\end{aligned}$$

Thus, n paths are described by R and this way, (G, R) generates

$$\begin{aligned}
{}_{\textit{III-n-path}}L(G,R) = \{&(a_1^{k_1+1}a_3^{k_3}\dots a_{m-1}^{k_{m-1}}a_m^{k_m}a_{m-2}^{k_{m-2}}a_{m-4}^{k_{m-4}}\dots a_2^{k_2})^{n+1} : \\
&k_i \geq 0, 1 \leq i \leq m\} \notin \mathbf{CF}.
\end{aligned}$$

9.3 Results

This section presents the results achieved in the investigation area defined in the previous section.

9.3.1 All Paths Restriction

Theorem 9.1. $\mathbf{CF} = \textbf{all-path-TC}(\mathbf{CF}, \mathbf{REG})$

Proof. Let $L \in \textbf{all-path-TC}(\mathbf{CF}, \mathbf{REG})$. We assume that L is generated by a tree controlled grammar, (G, R), where $G = (V, T, P, S)$ is a context-free grammar, R is a regular language over V, and (G, R) generates the language ${}_{\textit{all-paths}}L(G, R)$.

Next, we assume R is accepted by a deterministic finite automaton $M = (Q_M, V, R_M, s_M, F_M)$. Since the paths of a derivation tree of a context-free grammar are of the form xb with $x \in (V-T)^+, b \in T$, we assume that each $r \in R_M$ is of the form $pa \rightarrow q$ with either (a) $a \in V - T$ and $q \notin F_M$, or (b) $a \in T$ and $q \in F_M$.

Let G' be a context-free grammar $G' = (V', T, P', S')$, where $V' = Q \cup T$, $Q = \{\langle A, q_A\rangle : A \in V,\ q_A \in Q_M,\ qA \rightarrow q_A \in R_M \text{ for some } q \in Q_M\}$, $S' = \langle S, s_S\rangle, s_MS \rightarrow s_S \in R_M$, and P' is defined in the following way:

If

1. $A \rightarrow B_1B_2\dots B_n \in P$, $n \geq 1$;
2. $qA \rightarrow q_A \in R_M$, for some $q \in Q_M$;
3. $q_AB_i \rightarrow q_{B_i} \in R_M$, for each B_i, $i = 1, 2, \dots, n$;

then add $\langle A, q_A\rangle \rightarrow \overline{B_1B_2}\dots\overline{B_n}$ to P', where, for $i = 1, 2, \dots, n$,

if $B_i \in V - T$, then $\overline{B_i} = \langle B_i, q_{B_i} \rangle$ with $q_A B_i \to q_{B_i} \in R_M$,
if $B_i \in T$, then $\overline{B_i} = B_i$.

Without any loss of generality, we assume that $V \cap Q_M = \emptyset$. We define the function g from ${}_{G'}\triangle(y)$, $y \in (V')^*$, into ${}_{(G,R)}\triangle(x)$, $x \in V^*$, as

for all nodes labelled by $a \in T$, $g(a) = a$;
for all nodes labelled by $\langle A, q \rangle \in Q$, $g(\langle A, q \rangle) = A$.

To show that **all-path-TC**$(\mathbf{CF}, \mathbf{REG}) \subseteq \mathbf{CF}$, we first prove the next claim.
Claim: $t \in {}_{(G,R)}\triangle(x)$, $x \in V^*$, if and only if $d \in {}_{G'}\triangle(y)$, $y \in (V')^*$, such that $g(d) = t$.

Only-If Part: That is, if $t \in {}_{(G,R)}\triangle(x)$, $x \in V^*$, then $d \in {}_{G'}\triangle(y)$, $y \in (V')^*$, such that $g(d) = t$. This is established by induction on the number of the production trees, denoted by m, in $t \in {}_{(G,R)}\triangle(x)$, $x \in V^*$.

Basis: Let $m = 0$. Since $m = 0$ implies the zero-length derivation, the only production tree in t contains only the node S that corresponds to the start symbol of G. Clearly, the only production tree in d is the node that corresponds to the start symbol of G' – that is, $\langle S, s_S \rangle$ with $g(\langle S, s_S \rangle) = S$.

Induction Hypothesis: Suppose that the only-if part holds for all $t \in {}_{(G,R)}\triangle(uvw)$, $u, v, w \in V^*$, that contains m or fewer production trees, for some $m \geq 0$.

Induction Step: Consider any $t \in {}_{(G,R)}\triangle(uvw)$ that contains $m + 1$ production trees. Clearly, there is some subtree $t' \in {}_{(G,R)}\triangle(v)$ of t such that t' is a production tree.

Next, we remove just one production tree from t. If root$(t') = B$, then there is $t'' \in {}_{(G,R)}\triangle(uBw)$, where $u, w \in V^*, B \in V - T$, that contains m production trees. There is also $r : B \to v \in P$ (and its production tree $r_\triangle$) and t'' is a subtree of t. Hence, by the induction hypothesis, there is also $d'' \in {}_{G'}\triangle(y)$ such that $g(d'') = t''$.

Since $uBw \Rightarrow uvw\, [r]$ in G, $q_B B_i \to q_{B_i} \in R_M$, for each B_i in v, $i = 1, 2, \ldots, |v|$. Therefore, there is $r' \in P'$ (and its production tree $r'_\triangle$) such that $g(r'_\triangle) = r_\triangle$. Thus, we obtain $d \in {}_{G'}\triangle(y)$ with $g(d) = t$.

If Part: That is, if $d \in {}_{G'}\triangle(y)$, $y \in (V')^*$, then $t \in {}_{(G,R)}\triangle(x)$, $x \in V^*$, such that $g(d) = t$. This is established by induction on the number of the production trees, denoted by j, in $d \in {}_{G'}\triangle(y)$, $y \in (V')^*$.

Basis: Let $j = 0$. Since $j = 0$ implies the zero-length derivation, the only production tree in d contains only the node $\langle S, s_S \rangle$ that corresponds to the start symbol of G'. Clearly, the only production tree in t contains only the node S that corresponds to the start symbol of (G, R) and $g(\langle S, s_S \rangle) = S$.

Induction Hypothesis: Suppose that the if part holds for all $d \in {}_{G'}\triangle(uvw)$, $u, v, w \in (V')^*$, that contains j or fewer production trees, for some $j \geq 0$.

Induction Step: Consider any $d \in {}_{G'}\triangle(uvw)$ that contains $j + 1$ production trees. Clearly, there is some subtree $d' \in {}_{G'}\triangle(v)$ of d such that d' is a production tree.

Next, we remove just one production tree from d – that is, if root$(d') = \langle B, q \rangle$, then there is $d'' \in {}_{G'}\triangle(u\langle B, q \rangle w)$, where $u, w \in (V')^*, \langle B, q \rangle \in V' - T$, that contains j production trees. There is also $r : \langle B, q \rangle \to v \in P'$ (and its production tree $r_\triangle$) and d'' is a subtree of d. Hence, by the induction hypothesis, there is also $t'' \in {}_{(G,R)}\triangle(x)$ such that $g(d'') = t''$.

Since $u\langle B, q \rangle w \Rightarrow uvw\, [r]$ in G', there is $r' \in P$ (and its production tree $r'_\triangle$) such that $g(r'_\triangle) = r_\triangle$. For each $\langle B_i, q_{B_i} \rangle$ in v, there is some $q \in Q_M$ such that $q B_i \to q_{B_i} \in R_M, i = 1, 2, \ldots, |v|$. Thus, we obtain $t \in {}_{(G,R)}\triangle(x)$ with $g(d) = t$.

We can now easily obtain **all-path-TC**$(\mathbf{CF}, \mathbf{REG}) \subseteq \mathbf{CF}$ as follows.

- Let $t \in {}_{(G,R)}\triangle(x)$, with $x \in T^*$. Clearly $x \in L(G,R)$, there is $d \in {}_{G'}\triangle(y)$ such that $g(d) = t$, and $x = y \in L(G')$. Thus $L(G,R) \subseteq L(G')$.
- Let $d \in {}_{G'}\triangle(y)$, with $y \in T^*$. Clearly $y \in L(G)$, there is $t \in {}_{(G,R)}\triangle(x)$ such that $g(d) = t$, and $y = x \in L(G,R)$. Thus, $L(G') \subseteq L(G,R)$.

Therefore, $L(G,R) = L(G')$ and thus $\textbf{all-path-TC}(\mathbf{CF}, \mathbf{REG}) \subseteq \mathbf{CF}$.

Let L be a context-free language. Without any loss of generality, we assume that L is generated by a context-free grammar, $G = (V, T, P, S)$. Let (G', R) be a tree controlled grammar that generates ${}_{\textit{all-paths}}L(G', R) \in \textbf{all-path-TC}(\mathbf{CF}, \mathbf{REG})$, where $G' = G$, $R = (V - T)^+T$. Clearly $L(G) = L(G', R)$, therefore $\mathbf{CF} \subseteq \textbf{all-path-TC}(\mathbf{CF}, \mathbf{REG})$. Thus, $\textbf{all-path-TC}(\mathbf{CF}, \mathbf{REG}) = \mathbf{CF}$. □

Discussion, Notes, and Further Research Ideas

As a generalization of tree controlled grammars that generate the language under path-based control introduced in [80] (see Chap. 5), we have proved that the generative power of context-free grammars remains unchanged even if we restrict all paths in their derivation trees by regular languages.

9.3.2 Pumping Properties and Closure Properties

Theorem 9.2. *If $L \in \textbf{I-n-path-TC}(\mathbf{CF}, \mathbf{LIN})$, $n \geq 1$, then there are two constants, $k, q \geq 0$, such that each word, $z \in L$, with $|z| \geq k$ can be written as*

$$z = u_1 v_1 u_2 v_2 \dots u_{4n} v_{4n} u_{4n+1}$$

with $0 < |v_1 v_2 \dots v_{4n}| \leq q$ and for all $i \geq 1$, $u_1 v_1^i u_2 v_2^i \dots u_{4n} v_{4n}^i u_{4n+1} \in L$.

Proof. Let (G, R), where $G = (V, T, P, S)$ and $R \in \mathbf{LIN}$, be a tree controlled grammar that generates ${}_{\textit{I-n-path}}L(G, R)$. Without any loss of generality, we assume there is a linear grammar, $G' = (V', V, P', S')$, with $L(G') = R$.

Consider $t \in {}_{(G,R)}\triangle(z)$, for any $z \in {}_{\textit{I-n-path}}L(G, R)$. Clearly, for each path $p \in Q_t$, word$(p) = A_1 \dots A_\ell a$ with $A_1, \dots, A_\ell \in V - T$, for $\ell \geq 1$, and $a \in T$. Consider the productions $A_i \to x_i A_{i+1} y_i$, for $1 \leq i \leq \ell - 1$, used when passing from A_i to A_{i+1} on p and, corresponding to p, the production $A_\ell \to x_\ell a y_\ell$ used in the last step of the derivation in G.

Consider that any $x_i y_i$, $1 \leq i \leq \ell$, contains $B \in V - T$ that does not belong to any $p \in Q_t$. Clearly, there is a subword z' of z derived from B. Since $L(G) \in \mathbf{CF}$, then if $|z'| \geq k_1$, where $k_1 \geq 0$, then there are two subwords z'_1, z'_2 of z' that can be pumped and by Lem. 3.1, z'_1, z'_2 are bounded in length. See Fig. 9.3, (top-left).

If $L(G)$ is infinite, every $p \in Q_t$ is arbitrarily long, therefore also word$(p) \in L(G')$ is arbitrarily long. Thus, if word$(p) = u_p v_p x_p y_p z_p$ with $|u_p v_p x_p y_p z_p| \geq k_2$, for some $k_2 \geq 0$, then $u_p v_p x_p y_p z_p$ satisfies $u_p v_p^j x_p y_p^j z_p \in L(G')$, for $j \geq 1$. Hence, the derivations starting from the symbols of v_p and y_p can be repeated in G. Since (G, R) generates ${}_{\textit{I-n-path}}L(G, R)$, it follows that the derivations starting both from the symbols of v_p and y_p in G can be different for each $p \in Q_t$.

Consider the derivations starting from v_p and y_p in G. This leads to the pumping of four subwords of z—two in the left-hand side, two in the right-hand side corresponding to each $p \in Q_t$. Thus, we obtain $4n$ pumped subwords of z—denote them as v_{4m+1}, v_{4m+2},

v_{4m+3}, v_{4m+4}, $0 \le m \le n-1$. See Fig. 9.3, (top-right). By Lem. 3.1, the subwords v_1, v_2, $\dots$, v_{4n} are bounded in length, therefore $|v_1 v_2 \dots v_{4n}| \le q$. $\square$

Theorem 9.3. *If $L \in$ **III-n-path-TC(CF, LIN)**, $n \ge 1$, then there are two constants, $k, q \ge 0$, such that each each word, $z \in L$, with $|z| \ge k$ can be written as*

$$z = u_1 v_1 u_2 v_2 \dots u_{2n+2} v_{2n+2} u_{2n+3}$$

with $0 < |v_1 v_2 \dots v_{2n+2}| \le q$ and for all $i \ge 1$, $u_1 v_1^i u_2 v_2^i \dots u_{2n+2} v_{2n+2}^i u_{2n+3} \in L$.

Proof. Let (G, R), where $G = (V, T, P, S)$ and $R \in$ **LIN**, be a tree controlled grammar that generates ${}_{\textit{III-n-path}}L(G, R)$. Without any loss of generality, we assume there is linear a grammar, $G' = (V', V, P', S')$, with $L(G') = R$.

Consider $t \in {}_{(G,R)}\triangle(z)$, for any $z \in {}_{\textit{III-n-path}}L(G, R)$. Clearly, for each path $p \in Q_t$, word$(p) = A_1 \dots A_\ell a$ with $A_1, \dots, A_\ell \in V - T$, for $\ell \ge 1$, and $a \in T$. Consider the productions $A_i \to x_i A_{i+1} y_i$, for $1 \le i \le \ell - 1$, used when passing from A_i to A_{i+1} on p and, corresponding to p, the production $A_\ell \to x_\ell a y_\ell$ used in the last step of the derivation in G.

Consider that any $x_i y_i$, $1 \le i \le \ell$, contains $B \in V - T$ that does not belong to any $p \in Q_t$. Clearly, there is a subword z' of z derived from B. Since $L(G) \in$ **CF**, then if $|z'| \ge k_1$, where $k_1 \ge 0$, then there are two subwords z_1', z_2' of z' that can be pumped and by Lem. 3.1, z_1', z_2' are bounded in length. See Fig. 9.3, (top-left).

If $L(G)$ is infinite, every $p \in Q_t$ is arbitrarily long, therefore also word$(p) \in L(G')$ is arbitrarily long. Thus, if word$(p) = u_p v_p x_p y_p z_p$ with $|u_p v_p x_p y_p z_p| \ge k_2$, for some $k_2 \ge 0$, then $u_p v_p x_p y_p z_p$ satisfies $u_p v_p^j x_p y_p^j z_p \in L(G')$, for $j \ge 1$. Hence, the derivations starting from the symbols of v_p and y_p can be repeated in G. Since (G, R) generates ${}_{\textit{III-n-path}}L(G, R)$, it follows that the derivations starting from the symbols of v_p in G are common for all $p \in Q_t$, and the derivations starting from the symbols of y_p in G can be different for each $p \in Q_t$.

Consider the derivations starting from v_p in G. This leads to the pumping of two subwords v_1, v_{2n+2} of z—one in the left-hand side, one in the right-hand side corresponding to the common part of all $p \in Q_t$.

Consider the derivations starting from y_p in G. This leads to the pumping of two subwords of z—one in the left-hand side, one in the right-hand side corresponding to each $p \in Q_t$. Thus, we obtain $2n$ pumped subwords of z—denote them as v_{2m+2}, v_{2m+3}, $0 \le m \le n-1$. See Fig. 9.3, (bottom-left). By Lem. 3.1, the subwords v_1, v_2, $\dots$, v_{2n+2} are bounded in length, therefore $|v_1 v_2 \dots v_{2n+2}| \le q$. $\square$

Theorem 9.4. *If $L \in$ **V-n-path-TC(CF, LIN)**, $n \ge 1$, then there are two constants, $k, q \ge 0$, such that each word, $z \in L$, with $|z| \ge k$ can be written as*

$$z = u_1 v_1 u_2 v_2 u_3 v_3 u_4 v_4 u_5$$

with $0 < |v_1 v_2 v_3 v_4| \le q$ and for all $i \ge 1$, $u_1 v_1^i u_2 v_2^i u_3 v_3^i u_4 v_4^i u_5 \in L$.

Proof. Let (G, R), where $G = (V, T, P, S)$ and $R \in$ **LIN**, be a tree controlled grammar that generates ${}_{\textit{V-n-path}}L(G, R)$. Without any loss of generality, we assume there is a linear grammar, $G' = (V', V, P', S')$, with $L(G') = R$.

Consider $t \in {}_{(G,R)}\triangle(z)$, for any $z \in {}_{\textit{V-n-path}}L(G, R)$. Clearly, for each path $p \in Q_t$, word$(p) = A_1 \dots A_\ell a$ with $A_1, \dots, A_\ell \in V - T$, for $\ell \ge 1$, and $a \in T$. Consider the productions $A_i \to x_i A_{i+1} y_i$, for $1 \le i \le \ell - 1$, used when passing from A_i to A_{i+1} on p and, corresponding to p, the production $A_\ell \to x_\ell a y_\ell$ used in the last step of the derivation in G.

Consider that any $x_i y_i$, $1 \leq i \leq \ell$, contains $B \in V - T$ that does not belong to any $p \in Q_t$. Clearly, there is a subword z' of z derived from B. Since $L(G) \in \mathbf{CF}$, then if $|z'| \geq k_1$, where $k_1 \geq 0$, then there are two subwords z'_1, z'_2 of z' that can be pumped and by Lem. 3.1, z'_1, z'_2 are bounded in length. See Fig. 9.3, (top-left).

If $L(G)$ is infinite, every $p \in Q_t$ is arbitrarily long, therefore also word$(p) \in L(G')$ is arbitrarily long. Thus, if word$(p) = u_p v_p x_p y_p z_p$ with $|u_p v_p x_p y_p z_p| \geq k_2$, for some $k_2 \geq 0$, then $u_p v_p x_p y_p z_p$ satisfies $u_p v_p^j x_p y_p^j z_p \in L(G')$, for $j \geq 1$. Hence, the derivations starting from the symbols of v_p and y_p can be repeated in G. Since (G, R) generates ${}_{V\text{-}n\text{-}path}L(G, R)$, it follows that the derivations starting both from the symbols of v_p and y_p in G are common for all $p \in Q_t$.

Consider the derivations starting from v_p and y_p in G. This leads to the pumping of four subwords v_1, v_2. v_3, v_4 of z—two in the left-hand side, two in the right-hand side corresponding to the common part of all $p \in Q_t$. See Fig. 9.3, (bottom-right). By Lem. 3.1, the subwords v_1, v_2, v_3, v_4 are bounded in length, therefore $|v_1 v_2 v_3 v_4| \leq q$. □

Theorem 9.5. *For $n \geq 1$, $i \in \{$**I, II, III, IV, V**$\}$ and $T_G, T_R \in \{\mathbf{REG}, \mathbf{LIN}, \mathbf{CF}\}$, **n-path-TC**$(T_G, T_R)$, i-**n-path-TC**(T_G, T_R) are closed under intersection with regular languages, union, and non-erasing homomorphism.*

Proof. Let $n \geq 1$ and $L_1, L_2 \in$ **n-path-TC**(T_G, T_R). Assume tree controlled grammars (G_1, R_1), (G_2, R_2) where $G_1 = (V_1, T_1, P_1, S_1)$, $G_2 = (V_2, T_2, P_2, S_2)$, respectively.

Intersection with regular languages: It holds by the same argument as Prop. 1 in [81], where the construction is based on the evidence that **REG**, **LIN**, **CF** is closed under intersection with regular languages as well as under finite substitution on a linear language, and the independence on $n \geq 1$.

Union: Let (G, R) be a tree controlled grammar with $G = (V_1 \cup V_2 \cup \{S\}, T_1 \cup T_2, P_1 \cup P_2 \cup \{S \to S_1, S \to S_2\}, S)$, where $V_1 \cap V_2 = \emptyset$, $S \notin V_1 \cup V_2$, and $R = \{S\}R_1 \cup \{S\}R_2$. Obviously, ${}_{n\text{-}path}L(G, R) = {}_{n\text{-}path}L(G_1, R_1) \cup {}_{n\text{-}path}L(G_2, R_2)$. Analogously for i-**n-path-TC**(T_G, T_R), $i \in \{$**I, II, III, IV, V**$\}$.

Non-erasing homomorphism: Let $h : T_1^* \to (T_1{}')^+$ be a homomorphism. Let (G, R) be a tree controlled grammar with $G = (V_1, T_1{}', P \cup \{a \to h(a) : a \in T_1\}, S)$, where $T_1{}' = \bigcup_{a \in T_1} alph(h(a))$ and $R = \{paa' : pa \in R_1, a \in T_1, h(a) = xa'y, x, y \in T_1{}'^*, a' \in T'_1\}$. Assume $T_1 \cap T_1{}' = \emptyset$. Clearly, ${}_{n\text{-}path}L(G, R) = h(L_1) \in$ **n-path-TC**(T_G, T_R). Analogously for i-**n-path-TC**(T_G, T_R), where $i \in \{$**I, II, III, IV, V**$\}$. □

Theorem 9.6. *For $n \geq 1$,* **I-n-path-TC**, **III-n-path-TC**, *and* **V-n-path-TC** *are not closed under concatenation, intersection, and complement.*

Proof. Consider $n \geq 1$.

Concatenation: By Ex. 9.2, $L = \{a_1^j a_2^j \dots a_{2n+2}^j : j \geq 1\} \in$ **III-n-path-TC**, but by Lem. 9.3 $LL \notin$ **III-n-path-TC**. By Lem. 9.2 and Lem. 9.4 for the classes **I-n-path-TC** and **V-n-path-TC**, respectively, there is an analogous example.

Intersection: Let $L_1 = \{a_1^j a_2^j \dots a_{2n+2}^j a_{2n+3}^k : j, k \geq 1\}$, $L_2 = \{a_1^k a_2^j \dots a_{2n+2}^j a_{2n+3}^j : j \geq 1\}$. Clearly $L_1, L_2 \in$ **III-n-path-TC**. Obviously, by Lem. 9.3, $L_1 \cap L_2 = \{a_1^j a_2^j \dots a_{2n+3}^j : j \geq$

$1\} \notin$ **III-n-path-TC**. It can be proved analogously for **I-n-path-TC** and **V-n-path-TC**.

Complement: Since for $n \geq 1$, i-**n-path-TC**, where $i \in \{$**I, III, V**$\}$, are closed under union and not closed under intersection, it clearly follows from De-Morgan laws. □

Discussion, Notes, and Further Research Ideas

As one of the fundamental important language-class-characterizing properties, we have established pumping property for **I-n-path-TC**$(\mathbf{CF}, \mathbf{LIN})$, **III-n-path-TC**$(\mathbf{CF}, \mathbf{LIN})$, and **V-n-path-TC**$(\mathbf{CF}, \mathbf{LIN})$. These properties can be used exactly in the same way as well-known pumping properties for **REG**, **LIN**, **CF** to prove that a particular language does not belong to the given language class. The main open problem is to find out whether or not there are pumping properties also for **II-n-path-TC** and **IV-n-path-TC**.

9.3.3 Generative Power Approximation

Lemma 9.7. *Let $T_G, T_R \in \{\mathbf{REG}, \mathbf{LIN}, \mathbf{CF}\}$. Then $T_G \subseteq$ **path-TC**(T_G, T_R) and for $n \geq 1$, if $L \in$ **n-path-TC**(T_G, T_R), then for all $x \in L$ it holds $|x| \geq n$.*

Lemma 9.8. *For $T_G \in \{\mathbf{LIN}, \mathbf{CF}\}$ and $n \geq 1$, **n-path-TC**$(T_G, \mathbf{REG}) \subseteq T_G$.*

Proof. It follows from Th. 4.1 in [70] and Prop. 2 in [80]. Exactly the same technique is used to prove **n-path-TC**$(T_G, \mathbf{REG}) \subseteq T_G$. □

Lemma 9.9. *For $T_G \in \{\mathbf{LIN}, \mathbf{CF}\}$ and $n \geq 2$, **n-path-TC**$(T_G, \mathbf{REG}) \subset T_G$.*

Proof. The proper inclusion is obtained by combining Lem. 9.8 with 2nd statement of Lem. 9.7. □

Lemma 9.10. *For $n \geq 1$, $i \in \{$**I, II, III, IV, V**$\}$, and $T_G, T_R \in \{\mathbf{REG}, \mathbf{LIN}, \mathbf{CF}\}$,*

$$i\text{-}\mathbf{n\text{-}path\text{-}TC}(T_G, T_R) \subseteq \mathbf{n\text{-}path\text{-}TC}(T_G, T_R).$$

Proof. It follows from Def. 9.8 and Def. 9.9. For each $L \in i$-**n-path-TC**(T_G, T_R), $i \in \{$**I, II, III, IV, V**$\}$, there exists a tree controlled grammar (G, R) such that ${}_{n\text{-}path}L(G, R) = L$, $n \geq 1$. □

Theorem 9.11. *Depending on $n \geq 1$, there are infinite language hierarchies specified as $\bigcup_{i=1}^{n} j$-i-**path-TC**$(\mathbf{CF}, \mathbf{LIN})$, for $j \in \{$**I, III, V**$\}$.*

Proof. For $n \geq 1$, $\bigcup_{i=1}^{n}$ **III**-i-**path-TC**$(\mathbf{CF}, \mathbf{LIN}) \subset \bigcup_{i=1}^{n+1}$ **III**-i-**path-TC**$(\mathbf{CF}, \mathbf{LIN})$ by Lem. 9.3. Analogously by Lem. 9.2 for $\bigcup_{i=1}^{n}$ **I**-i-**path-TC**$(\mathbf{CF}, \mathbf{LIN})$ and by Lem. 9.4 for $\bigcup_{i=1}^{n}$ **V**-i-**path-TC**$(\mathbf{CF}, \mathbf{LIN})$. □

Theorem 9.12. *For every $n \geq 1$ and $T_G, T_R \in \{\mathbf{REG}, \mathbf{LIN}, \mathbf{CF}\}$, for each $L \in T_G$, there are*

$$L_1 \in \mathbf{n\text{-}path\text{-}TC}(T_G, T_R) \text{ and } L_2 \in \mathbf{FIN} \text{ such that } L \subseteq L_1 \cup L_2.$$

Proof. Assume $G = (V, T, P, S)$ with $G \in \mathbb{G}_{T_G}$ and $L(G) = L$. Let (G, R) be a tree controlled grammar where $R \subseteq (V - T)^*T$ and $n \geq 1$. By 2nd statement of Lem. 9.7, for each $z \in L$ with $|z| \geq n$, $z \in {}_{n\text{-}path}L(G, R)$. Consider $L_f = \{z \in L : |z| < n\} \in \mathbf{FIN}$. Clearly, $L \subseteq {}_{n\text{-}path}L(G, R) \cup L_f$. □

Theorem 9.13. *Let $L \in$ **n-path-TC(CF, CF)**, for $n \geq 1$. Then, there exists*

$$L' = L\{\blacklozenge^q\} \in \mathbf{PSC} \text{ for } q \geq 2n+1.$$

Proof. First, consider the following construction and then its explanation. Let (G, R) be a tree controlled grammar that generates ${}_{n\text{-}path}L(G, R)$, where $G = (V, T, P, S)$, $N = V - T$, and R is a control language over V. Without any loss of generality, we assume a context-free grammar $G' = (V', V, P', S')$ with $L(G') = R \subseteq \{S\}N^*T$. Consider the following definitions:

$N_{\bullet\circ} = \{\bullet_i, \circ_i : 1 \leq i \leq n\}$,
$N_{\overline{X}} = \bigcup_{i=1}^{n}\{\overline{X_i} : X \in V\}$,
$N_{\widehat{X}} = \bigcup_{i=1}^{n}\{\widehat{X_i} : X \in V\}$,
$P_{ST} = \{(S'' \to \overline{S}\blacktriangle \circ_1 S'_1 \circ_2 S'_2 \cdots \circ_n S'_n)\}$,
$P_G = \{(r) : r \in P\}$,
$P_{G'} = \bigcup_{i=1}^{n}\{(X_i \to h(x_{i,1} \dots x_{i,m})) : X \to x_1 \dots x_m \in P', m \geq 1\}$,
$P_{CMN} = \{(\overline{A} \to u_1\overline{X}u_2, \blacktriangle \to \blacktriangle, \circ_1 \to \bullet_1, \widehat{A_1} \to \circ_1, \dots, \circ_n \to \bullet_n, \widehat{A_n} \to \circ_n) :$
$\quad A \to u_1Xu_2 \in P, X \in V\}$,
$P_{MID} = \{(\overline{A} \to u_1\overline{X_1}u_2\overline{X_2}u_3 \dots u_n\overline{X_n}u_{n+1}, \blacktriangle \to \blacktriangledown,$
$\quad \circ_1 \to \bullet_1, \widehat{A_1} \to \circ_1, \dots, \circ_n \to \bullet_n, \widehat{A_n} \to \circ_n) :$
$\quad A \to u_1X_1u_2X_2u_3 \dots u_nX_nu_{n+1} \in P, X_1, \dots, X_n \in V\}$,
$P_{UNQ} = \bigcup_{i=1}^{n}\{(\overline{A} \to u\overline{X}v, \blacktriangledown \to \blacktriangledown, \circ_i \to \bullet_i, \widehat{A_i} \to \circ_i) : A \to uXv \in P, X \in V\}$,
$P_T = \bigcup_{i=1}^{n}\{(\overline{a} \to a, \blacktriangledown \to \blacktriangledown, \circ_i \to \bullet_i, \widehat{a_i} \to \bullet_i) : a \in T\}$,
$P_f = \{\blacktriangledown \to \blacklozenge\} \cup \{\bullet_i \to \blacklozenge : 1 \leq i \leq n\}$,

where $N' = V' - V$ and h is the homomorphism from $(N' \cup V)^*$ to $(N' \cup N_{\widehat{X}})^*$ defined by:

if $X \in N'$: $h(X) = X$;
if $X \in V$: $h(X) = \widehat{X}$.

Without any loss of generality, we assume that N, N', $N_{\bullet\circ}$, $N_{\overline{X}}$, and $N_{\widehat{X}}$ are pairwise disjoint. Let $G'' = (V'', T \cup \{\blacklozenge\}, P'', S'')$ be a propagating scattered context grammar with

$V'' = N \cup N_{\bullet\circ} \cup N' \cup N_{\overline{X}} \cup N_{\widehat{X}} \cup T \cup \{S'', \blacktriangle, \blacktriangledown, \blacklozenge\}$,
$P'' = P_{ST} \cup P_G \cup P_{G'} \cup P_{CMN} \cup P_{MID} \cup P_{UNQ} \cup P_T$.

Explanation: For better readability, note that the symbols of $N_{\bullet\circ}$ are used to indicate left position in each controlled path, $N_{\overline{X}}$ are used as nonterminal on the controlled paths, $N_{\widehat{X}}$ are used for a control language, the production in P_{ST} is the only production with the start symbol on the left-hand side, the production s in P_G are used to rewrite beyond controlled paths, $P_{G'}$ generates generate a control language, P_{CMN} rewrite in a common part of the paths, P_{MID} rewrite the last common node of controlled paths, P_{UNQ} rewrite in the unique parts of controlled paths, P_T generate terminals, and P_f finish a derivation.

First, G'' generates $x = \overline{S}\blacktriangle \circ_1 S'_1 \circ_2 S'_2 \cdots \circ_n S'_n$. Obviously, from each symbol S'_i, for $1 \leq i \leq n$, by $r \in P_{G'}$, G'' constructs the word z_i of the form $\widehat{A_{i,1}}\widehat{A_{i,2}} \dots \widehat{A_{i,k}}\widehat{a_i}$ where $A_{i,1}A_{i,2} \dots A_{i,k}a_i \in L(G')$, for some $k \geq 1$. Observe that each z_i is constructed nondeterministically. Thus, we obtain the word of the form $\overline{S}\blacktriangle \circ_1 z_1 \circ_2 z_2 \cdots \circ_n z_n$, where z_i corresponds to the i-th controlled path, for $1 \leq i \leq n$. Then, G'' continues in a derivation $\overline{S}\blacktriangle \circ_1 z_1 \circ_2 z_2 \cdots \circ_n z_n \Rightarrow^* w\blacklozenge\blacklozenge^{|z_1|}\blacklozenge^{|z_2|} \dots \blacklozenge^{|z_n|}$ where $w \in T^*$ in the following way (see the illustration on Fig. 9.4):

1. Indicated by $\blacktriangle$, G'' simulates the derivation from $A \in N$ on the common part of all controlled paths using $r \in P_{CMN}$ to (i) generate a symbol $\overline{B} \in N_{\overline{X}}$, (ii) rewrite $\blacktriangle$ to $\blacktriangle$, (iii) for each $1 \leq i \leq n$, rewrite $\circ_i$ to $\bullet_i$ and replace one occurrence of $\widehat{A_i} \in N_{\widehat{X}}$ with $\circ_i$.
2. G'' simulates the derivation starting from $A \in N$ in which all controlled paths are divided—in the last common symbol of all controlled paths. That is, G'' uses $r \in P_{MID}$ to (i) generate n symbols $\overline{B} \in N_{\overline{X}}$, (ii) replace $\blacktriangle$ by $\blacktriangledown$, (iii) for each $1 \leq i \leq n$, rewrite $\circ_i$ to $\bullet_i$ and replace one $\widehat{A_i} \in N_{\widehat{X}}$ with $\circ_i$.
3. Indicated by $\blacktriangledown$, G'' simulates the derivation starting from $A \in N$ in the unique part of each controlled path. That is, G'' uses $r \in P_{UNQ}$ to (i) generate a symbol $\overline{B} \in N_{\overline{X}}$, (ii) rewrite $\blacktriangledown$ to $\blacktriangledown$, (iii) for some i, $1 \leq i \leq n$, rewrite $\circ_i$ to $\bullet_i$ and replace $\widehat{A_i} \in N_{\widehat{X}}$ with $\circ_i$.
4. G'' replaces all occurrences of $\bullet_i$ and $\blacktriangledown$ by $\blacklozenge$.
5. G'' uses $r \in P_G$ to simulate every derivation starting from each $A \in N$ that belongs to no controlled path.

Note that the grammar G'' works in the leftmost way—this is done by using the symbols $\bullet_i$ and $\circ_i$, $1 \leq i \leq n$, that ensure that any skipped symbol from $N_{\widehat{X}}$ cannot be rewritten anymore. Clearly, the derivation is finished only after removing all symbols $\widehat{A_i} \in N_{\widehat{X}}$, for all $1 \leq i \leq n$. In this way, we obtain a word of the form $x = w\blacklozenge\blacklozenge^{|z_1|}\blacklozenge^{|z_2|}\ldots\blacklozenge^{|z_n|}$ where $w \in L$ and the derivation tree contains n paths described by R. Thus, the number of $\blacklozenge$s in x is equal to the length of all controlled paths plus 1—i.e., $|w| \geq 2n+1$ (since each controlled path is of the length at least 2).

Note that the construction can be easily modified for $n = 1$ where the division node is omitted and thus, the markers $\blacktriangle$ and $\blacktriangledown$ are not needed anymore.

□

Corollary 9.14. *Let $L \in$ **n-path-TC(CF, CF)**, for $n \geq 1$. Then, there exists $L' \in$ **PSC** with $L = h(L')$.*

Proof. By Th. 9.13, for each $L \in$ **n-path-TC(CF, CF)**, for $n \geq 1$ there is $L' = L\{\blacklozenge^q\} \in$ **PSC** for $q \geq 2n+1$. Consider a tree controlled grammar (G, R) with ${}_{n\text{-}path}L(G, R) = L$ where $G = (V, T, P, S)$ and PSC grammar $G' = (V', T, P', S)$ with $L(G') = L'$. Let $h : V^* \to (V')^*$ be the morphism defined by $h(\blacklozenge) = \varepsilon$, otherwise $h(A) = A$. Clearly, $L = h(L')$. □

Discussion, Notes, and Further Research Ideas

Based on the aforementioned counterargument against the generative power of path controlled grammars (see Sec. 8.3.4), we have tried to establish the generative power for n-path controlled grammar. Despite all our effort so far, we were able to find only its approximation. However, this approximation does not say too much since it is well-known that **PSC** is not closed under erasing homomorphism (see Col. in 3.18 in [92]), thus we have informally concluded that: *„We either have the power to check what we need but not to remove it (using* **PSC***) or vice versa (using* **MAT***).“*

9.3.4 Syntax Analysis

Theorem 9.15. *For a tree controlled grammar, (G, R) with an unambiguous context-free grammar, G, and a linear control language, R, the membership $x \in {}_{nc\text{-}n\text{-}path}L(G, R)$, $n \geq 1$, is decidable in $O(|x|^k)$, for some $k \geq 0$.*

Proof. We assume R is generated by some unambiguous linear grammar. Since G is unambiguous, it is well-known that we can decide whether or not $x \in L(G)$ in $O(|x|^2)$. We distinguish two cases. Clearly, if $x \notin L(G)$, then $x \notin {}_{nc\text{-}n\text{-}path}L(G, R)$. If $x \in L(G)$, since G is unambiguous, we can construct unique derivation tree, t, of $x \in L(G)$ in $O(|x|^2)$. Since each path of t ends in a leaf, t contains $|x|$ paths. Clearly, the height of t is polynomially bounded by some $l \geq 2$ with respect to $|x|$. Thus, for any $x \in L(G)$, the length of each path p of t and therefore also $|\operatorname{word}(p)|$ are bounded by l. Because R is unambiguous and $|\operatorname{word}(p)| \leq l$ for each $p \in Q_t$, it is well-known that we can decide whether or not $\operatorname{word}(p) \in L(R)$ in a polynomial time. If for at least n paths, $p_1, p_2, \ldots, p_n$, of derivation tree of x in G, $\operatorname{word}(p_i) \in R$ holds, for $i \in 1, 2, \ldots, n$, then $x \in {}_{nc\text{-}n\text{-}path}L(G, R)$. $\square$

As it straightforwardly follows from Th. 9.15, the proposed way to solve the membership problem leads to a parsing method working, in essence, in two phases: (1) construction of a derivation tree, t, of x in G by top-down parsing method, and (2) checking that at least $n \geq 1$ paths of t belong to R. However, from the practical viewpoint, the situation may occur in which during the phase (1) above we already know that currently constructed derivation tree cannot contain the required number of paths described by the words of R. Informally, we do not have to wait with starting the phase (2) until the phase (1) is completely done (i.e., until t is completely constructed).

Top-Down Parsing of nc-n-path-TC(CF, LIN)

Consider ${}_{nc\text{-}n\text{-}path}L(G, R)$, for some $n \geq 1$, as the language of a tree controlled grammar (G, R) where $G = (V, T, P, S)$ is an unambiguous context-free grammar. We assume R is generated by an unambiguous context–free grammar, $G_R = (V_R, V, P_R, S_R)$. Adjust the idea behind Th. 9.15 as follows.

We construct a labelled derivation tree with the set of labels $\Psi = \{0, 1\}$ and the following semantics. Let p be a path of derivation tree, t, in G and e be an edge between any two consecutive nodes of p. Then, label $0 \in \Psi$ of any e of p represents p is not described by R (i.e., $\operatorname{word}(p) \notin R$). Label $1 \in \Psi$ of all e of p represents p can potentially be described by R.

Consider that for the decision whether or not $x \in L(G)$, we use the well-known top-down parsing method to construct a derivation tree, t, of x in G. Let us suppose that a production, $r : A \to A_1A_2 \ldots A_j \in P$, $j \geq 1$, is used in the derivation step $X \Rightarrow Y$, $X, Y \in V^*$ in G. In addition, we need to determine the value of the labels of the edges between A and each A_j, for $j = 1, 2, \ldots, n$, related to the application of r. Let t' be a derivation tree that corresponds to the derivation $S \Rightarrow^* w_1A_1A_2 \ldots A_jw_2$ in G, for some $w_1, w_2 \in V^*$. Essentially, t' is a subtree of t. Clearly, each path of t' is the beginning part of at least one path in t. Next, we distinguish the following cases: if all the edges of t' are labelled, we can proceed to next derivation step in G; if some of the edges in t' are not labelled, we need to compute the values of missing labels.

For each unlabelled edge, e, of a path, p', in t', we check whether or not G_R can generate the word of the form $\operatorname{word}(p')w$ with $w \in (V_R - V)^*T \cup \{\varepsilon\}$. Since $|\operatorname{word}(p')|$ is finite, it can be checked in a polynomial time. If so, we add label $1 \in \Psi$ to edge e; otherwise, we add label $0 \in \Psi$ to edge e. Note that this phase can be optimized in such a way that we do the test whether or not G_R can generate word$(p')w$ with $w \in (V_R - V)^*T \cup \{\varepsilon\}$ symbol-by-symbol during the generation of word(p') in G_R. Next, we distinguish the following cases: if t' contains no leaf with input edge labelled by 1, then $x \notin {}_{\textit{nc-n-path}}L(G, R)$; if t' contains at least one leaf labelled by symbol of $V - T$, we proceed to the next derivation step in G; or if all the leafs of t' are labelled by the symbols of T and for at least n of the leafs of t, there is an input edge labelled by 1, then $x \in {}_{\textit{nc-n-path}}L(G, R)$.

Thus, it is possible to check whether or not the paths of derivation tree t of LL context-free grammar can potentially be described by given unambiguous context–free language already during the building of t by LL parser. The following example explains the syntax analysis in more detail.

Example 9.5. Consider the tree controlled grammar (G, R) that generates ${}_{\textit{nc-n-path}}L(G, R)$ where

$$\begin{aligned}
&G = (\{S,\ A,\ B,\ a,\ b,\ c,\ d,\ e,\ k\},\ \{a, b, c, d, e, k\},\ P,\ S),\\
&P = \{1:\ S \to AA, \qquad 2:\ A \to aAd, \qquad 3:\ A \to bBc,\\
&\qquad 4:\ A \to e, \qquad 5:\ B \to bBc, \qquad 6:\ B \to k\},\\
&R = \{SA^mB^{m-1}k:\ m \geq 1\}.
\end{aligned}$$

Obviously, $L(G) = \{a^i(b^jkc^j{+}e)d^ia^s(b^tkc^t{+}e)d^s:\ i, j, t, s \geq 0\}$. Clearly, ${}_{\textit{nc-1-path}}L(G, R) = L(G)$ with $i = j$ or $s = t$, and ${}_{\textit{nc-2-path}}L(G, R) = L(G)$ with $i = j = s = t$. The LL table for G is constructed by the well-known algorithm (see [112]) and it is presented in Tab. 9.1.

	a	b	c	d	e	k	$
S	1	1			1		
A	2	3			4		
B		5				6	

Table 9.1: LL table for grammar G from Ex. 9.5.

Idea: The syntax analyser uses two pushdown automata and a list of 3-tuples containing state of the second automaton, contents of its pusdown, and a pointer to the first-automaton's pushdown. The first pushdown automaton simulates the construction of a derivation tree by the LL table in the well-known way. More precisely, if the top-most symbol on the pushdown is a non-terminal A, the first input symbol is a, and there is a production $A \to x$ on position $[A, a]$ in the LL table, then the automaton rewrites A on the pushdown by reversal(x) (expansion step); if a on the pushdown's top is a terminal and a is the first input symbol, the automaton reads a from the input and removes a from the pushdown (comparative step); and other cases represent a syntax error.

Let us return to the example and consider the input word *abkcdaed*. In the beginning, the top-most symbol on the pushdown of the first automaton is S. Since a is the first input symbol and there is production 1 on the position $[S, a]$ in the table, the automaton rewrites S by AA on its pushdown.

During the computation, the second automaton is checking the potentially valid paths in the derivation tree. At the beginning, the list contains one item $(q_0, \widehat{S}, 1)$ where 1 represents the first position on the first-automaton's pushdown from the bottom and $\widehat{S}$ is the start pushdown symbol of the second automaton. If the first automaton makes a computation step with symbol a on the pushdown's top and there is a tuple (q, α, p) in the list where p

is the pointer to the symbol, the second automaton places α onto its pushdown, automaton moves to q and it makes the moves for a as the first input symbol until it does not need next input symbol. For example, after the expansion from S to AA, the second automaton finds list-item $(q_0, \widehat{S}, 1)$ and it moves to state q_0 and places $\widehat{S}$ onto the pushdown.

Then, it simulates the move $\widehat{S}q_0S \vdash \beta q$ where $\widehat{S}q_0S \to \beta q$ is a computatinal rule of the second automaton. If $a = A$ was a non-terminal and first automaton made expansion by production $A \to x$, then the syntax analyser removes the used list-item and if the second automaton did not reject the input, then there are l tuples (q, β, i) inserted into the list, for all $i = t+1, \ldots, t+l$ with $l = |x|$, top-most-symbol-position t before the expansion, and β as the current content of the pushdown. Otherwise, for a as a terminal symbol, the comparison is done. If the second automaton accepts, the pointer of the used tuple is rewritten to 0 where 0 denotes accepted path. Assuming that it does not accept with a terminal, a, the tuple is removed from the list.

Consider the input word *abkcdaed* again. For some definition of the second automaton, the syntax analysis proceeds as it is given in Tab. 9.2.

1. PDA	2. PDA	Pointer	Tuples
$Sqabkcdaed$	q_0	$(q_0, \varepsilon, 1)$	$(q_0, \varepsilon, 1)$
$AAqabkcdaed$	q_0	$(q_0, \varepsilon, 1)$	$(q_1, \varepsilon, 1)$, $(q_1, \varepsilon, 2)$
$AdAaqabkcdaed$	q_1	$(q_1, \varepsilon, 2)$	$(q_1, \varepsilon, 1)$, $(q_1, A, 2)$, $(q_1, A, 3)$, $(q_1, A, 4)$
$AdAqbkcdaed$	Aq_1	$(q_1, A, 4)$	$(q_1, \varepsilon, 1)$, $(q_1, A, 2)$, $(q_1, A, 3)$
$AdcBbqbkcdaed$	Aq_1	$(q_1, A, 3)$	$(q_1, \varepsilon, 1)$, $(q_1, A, 2)$, $(q_1, AA, 3)$, $(q_1, AA, 4)$, $(q_1, AA, 5)$
$AdcBqkcdaed$	AAq_1	$(q_1, AA, 5)$	$(q_1, \varepsilon, 1)$, $(q_1, A, 2)$, $(q_1, AA, 3)$, $(q_1, AA, 4)$
$Adckqkcdaed$	AAq_1	$(q_1, AA, 4)$	$(q_1, \varepsilon, 1)$, $(q_1, A, 2)$, $(q_1, AA, 3)$, $(q_1, A, 4)$
$Adcqcdaed$	Aq_1	$(q_1, A, 4)$	$(q_1, \varepsilon, 1)$, $(q_1, A, 2)$, $(q_1, AA, 3)$, $(q_1, A, 0)$
$Adqdaed$	AAq_1	$(q_1, AA, 3)$	$(q_1, \varepsilon, 1)$, $(q_1, A, 2)$, $(q_1, A, 0)$
$Aqaed$	Aq_1	$(q_1, A, 2)$	$(q_1, \varepsilon, 1)$, $(q_1, A, 0)$
$dAaqaed$	q_1	$(q_1, \varepsilon, 1)$	$(q_1, A, 0)$, $(q_1, A, 1)$, $(q_1, A, 2)$, $(q_1, A, 3)$
$dAqed$	Aq_1	$(q_1, A, 3)$	$(q_1, A, 0)$, $(q_1, A, 1)$, $(q_1, A, 2)$
$deqed$	Aq_1	$(q_1, A, 2)$	$(q_1, A, 0)$, $(q_1, A, 1)$, $(q_1, AA, 2)$
dqd	AAq_1	$(q_1, AA, 2)$	$(q_1, A, 0)$, $(q_1, A, 1)$
q	Aq_1	$(q_1, A, 1)$	$(q_1, A, 0)$

Table 9.2: Parsing of *abkcdaed* corresponding to tree controlled grammar (G, R) from Ex. 9.5.

As it can be seen, only one item with 0 in the last component remains in the list—there is one path belonging to the control language. If we require $n \geq 2$ paths described by the control language, the input word is not accepted.

Bottom-Up Parsing of **nc-n-path-TC**$(\mathbf{CF},\mathbf{LIN})$

The previous section deals in principle with top-down parsing method (LL parser). However, the weakness of LL parser is the assumption that a context-free grammar is unambiguous. Moreover, the method demonstrated in Ex. 9.5 assumes that the grammar is LL. Essentially, the same idea is applicable also on bottom-up parsing methods (e.g., LR parser) which can handle a larger range of the languages. Therefore, we briefly discuss the ideas of parsing methods for **nc-n-path-TC**$(\mathbf{CF},\mathbf{LIN})$ in terms of LR parsing (see [4] and [89] for the definition of LR property).

One of the advantages of bottom-up parsers is that we do not need to require that in a tree controlled grammar, (G, R), G is LL grammar. On the other hand, concerning the bottom-up parsing, we have to deal with the ambiguity. However, it is well-known that the question whether or not a context-free grammar is ambiguous is undecidable. Indeed, the problem can be reduced to the Post Correspondence Problem which is well-known to be undecidable (see [105]).

It is also well-known that for some ambiguous context-free grammars, there exists equivalent context-free grammar which is unambiguous. The ambiguity of a context-free grammar can be restricted basically by removing the unit productions. We assume that a context-free grammar contains only usable productions—only those productions, which can be used during the derivation. Clearly, if $G = (V, T, P, S)$ is a context-free grammar with $r : A \to A \in P$, for some $A \in V - T$, then G is ambiguous since r can be used during the derivation of $x \in L(G)$ arbitrarily many times and thus generate arbitrarily many different derivation trees for x in G.

Obviously, since the unit productions generate nothing, they can be removed from a context-free grammar G without affecting $L(G)$. However, removing the unit productions from G in a tree controlled grammar (G, R) affects the paths of the derivation trees of $x \in L(G)$. Thus the identity ${}_{\textit{nc-n-path}}L(G, R) = {}_{\textit{nc-n-path}}L(G', R)$, where G' is obtained by removing the unit productions from G, does not hold. However, the equivalence $L(G) = L(G')$ holds.

Theorem 9.16. *For a tree controlled grammar, (G, R), where G is a context-free grammar and $R \in$ **LIN**, there is a tree controlled grammar, (G', R'), such that G' does not contain unit productions and ${}_{\textit{nc-n-path}}L(G, R) = {}_{\textit{nc-n-path}}L(G', R')$, $n \geq 1$.*

Proof. Consider $G = (V, T, P, S)$ and let G' be a context-free grammar obtained from G by removing the unit productions. Therefore, G' can be constructed by well-known algorithm in a polynomial time (see 5.1.3.3 in [89] used for transformation of a context-free grammar to an equivalent context-free grammar without unit productions). Thus, we get $G' = (V, T, P', S)$ such that for all $x \in L(G')$, there is no derivation in G' of the form $B \Rightarrow^* A$, for some $A, B \in V - T$.

The paths in the derivation trees of G' are described by the words of the form $(V - T)^*T$. Basically, we need to read such a words and remove such symbols $A \in V - T$ which corresponds to the application of $B \to A \in P$ in G. This is done by gsm mapping M such that M reads the words s of the form $(V - T)^*T$ and nondeterministically removes or lets unchanged each symbol $A \in (V - T)$ with $B \to A \in P$ and BA is subword of s. Since **LIN** is closed under gsm mappings (see [25]), also $M(R) \in$ **LIN**. This way, we get $M(R)$ with $M(R) \neq R$. However, ${}_{\textit{nc-n-path}}L(G, R) = {}_{\textit{nc-n-path}}L(G', M(R))$. □

Consider a tree controlled grammar, (G, R), and let, (G', R'), be constructed as described above. Clearly, G' does not need to be unambiguous since the unit productions are not the only cause of the ambiguity. Consider, however, any $x \in {}_{\textit{nc-n-path}}L(G', R')$. Obviously, there is a derivation tree t of x in G'. Since there are no unit productions in G' and for each $x \in L(G')$, $|x|$ is finite, the height of t with respect to $|x|$ is bounded by $log\,|x|/log\,2$. Thus there is at most m, for some $m \geq 1$, derivation trees of x in G' and G' is m-ambiguous.

Theorem 9.17. *For a tree controlled grammar, (G, R), where $G = (V, T, P, S)$ is m-ambiguous LR grammar, $m \geq 1$, and an unambiguous language $R \in$ **LIN** , the membership $x \in {}_{\textit{nc-n-path}}L(G, R)$, $n \geq 1$, is decidable in $O(|x|^k)$, for some $k \geq 0$.*

Proof. If $x \in L(G)$, then we can construct at most m derivation trees of $x \in L(G)$ in $O(m.|x|^2)$ by LR parser. Then, if for at least j paths, $p_1, p_2, \ldots, p_j$, of at least one derivation tree of x in G it holds that $\mathrm{word}(p_i) \in R$ for $i \in 1, 2, \ldots, j$, then $x \in {}_{\textit{nc-n-path}}L(G, R)$.

Hence, the syntax analysis of **nc-n-path-TC**(**CF**, **LIN**) with LR grammar G and unambiguous linear control language can be done in a polynomial time also in the case of LR parsing. □

Discussion, Notes, and Further Research Ideas

We have demonstrated that for $L \in$ **nc-n-path-TC**(**CF**, **LIN**) under the assumption that L is generated by a tree controlled grammar, (G, R), in which both G as well as R are unambiguous and, furthermore, G is restricted to be LL grammar, there is parsing method working in a polynomial time. This method can check whether or not the paths of a derivation tree, t, of $x \in L(G)$ belong to control language, R, in the time of building t. Moreover, when we consider LR parser for $L \in$ **nc-n-path-TC**(**CF**, **LIN**) under assumption that L is generated by tree controlled grammar, (G, R), in which G has bounded ambiguity (i.e., G is unambiguous or m-ambiguous) and an unambiguous language, $R \in$ **LIN**, there is also a parsing method working in a polynomial time.

However, the open question is whether or not there is a polynomial time parsing method if G is not LL or if G is ambiguous. It is also of interest to quantify the worst case of the parsing complexity more precisely.

The open investigation area is represented by the transformation of n-path tree controlled grammars into some Chomsky-like normal forms which would lead to the possibility to use general parsing methods based on Chomsky normal form.

Concerning the parsing methods for tree controlled grammars (and also similar rewriting system), there is great possibility to use n-Accepting Restricted Pushdown Automata Systems, which deals with the sets of n-tuples of words. These systems are studied in [15], [16], and [18].

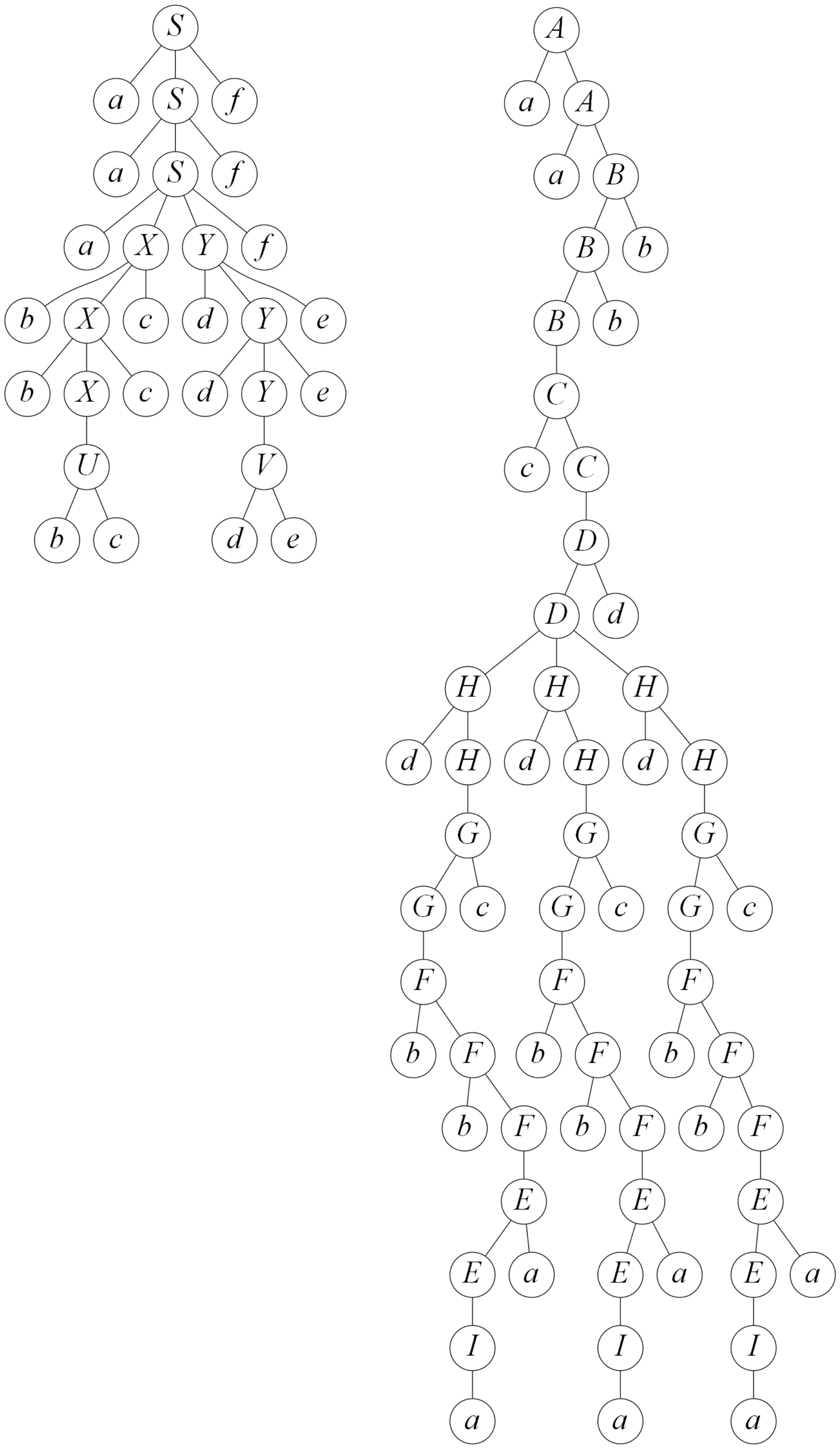

Figure 9.2: Illustration of the derivations of $a^3b^3c^3d^3e^3f^3$ with two paths of the form $S^nX^nUb \cup S^nY^nVd$, where $n \geq 1$, in Ex. 9.1 (left) and $(a^2cdb^2)^4$ with three paths of the form $A^rB^sC^tD^uH^uG^tF^sE^rIa$, where $r, s, t, u \geq 0$, in Ex. 9.3 (right).

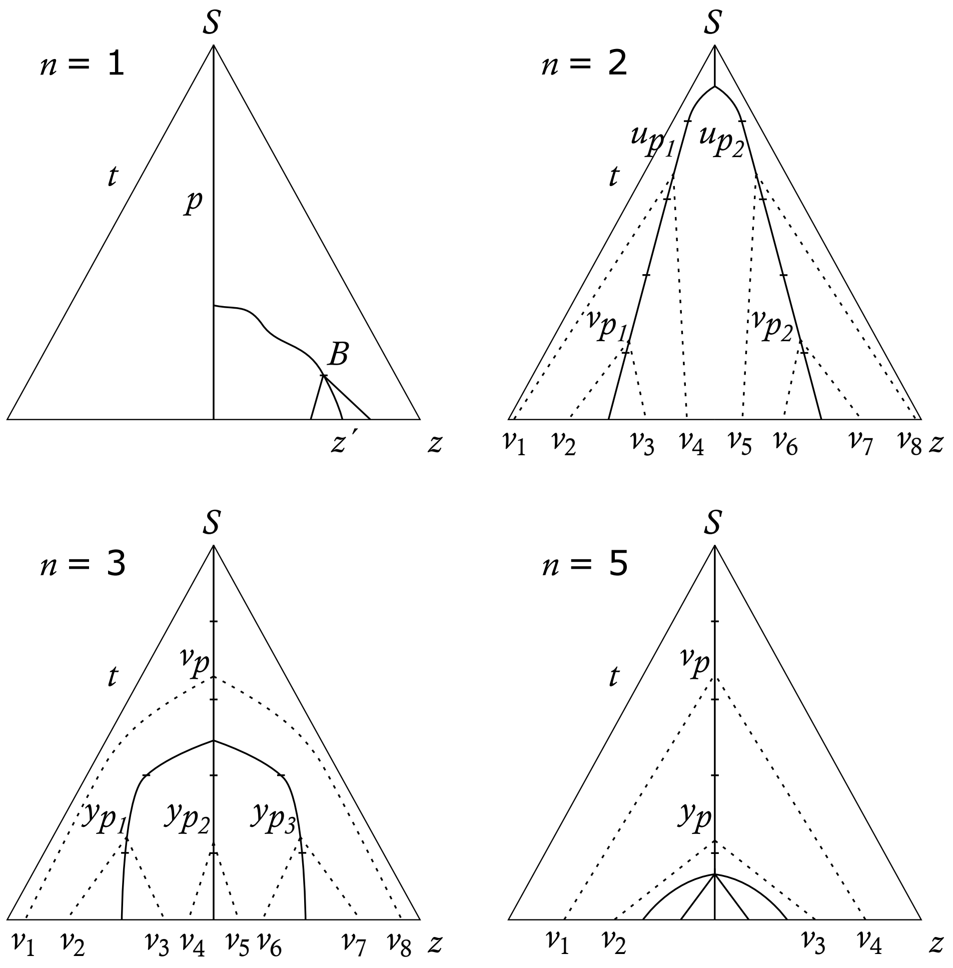
S
n = 1
t
p
B
z′
z
S
n = 2
u_{p_1}
u_{p_2}
t
v_{p_1}
v_{p_2}
v_1 v_2 v_3 v_4 v_5 v_6 v_7 v_8 z
S
n = 3
t
v_p
y_{p_1}
y_{p_2}
y_{p_3}
v_1 v_2 v_3 v_4 v_5 v_6 v_7 v_8 z
S
n = 5
t
v_p
y_p
v_1 v_2 v_3 v_4 z

Figure 9.3: Illustration of the idea behind the proof of Th. 9.2, Th. 9.3, and Th. 9.4. On all four parts, there is an example of a derivation tree of word z in a tree controlled grammar (G, R), where $G = (V, T, P, S)$, for given $n \geq 1$: (top-left) a derivation starting from $B \in N$, where B does not belong to any controlled path, leads to a derivation of subword z' of z; (top-right) a derivation of $z \in {}_{I\text{-}n\text{-}path}L(G, R)$, where there are two different iterative parts for each controlled path, leads to a derivation of $4n$ iterative parts of z; (bottom-left) a derivation of $z \in {}_{III\text{-}n\text{-}path}L(G, R)$, where there is one common iterative part for all controlled paths and one different iterative part for each controlled path, leads to a derivation of $2n + 2$ iterative parts of z; and (bottom-right) a derivation of $z \in {}_{V\text{-}n\text{-}path}L(G, R)$, where there are two iterative parts common for all controlled paths, leads to a derivation of four iterative parts of z.

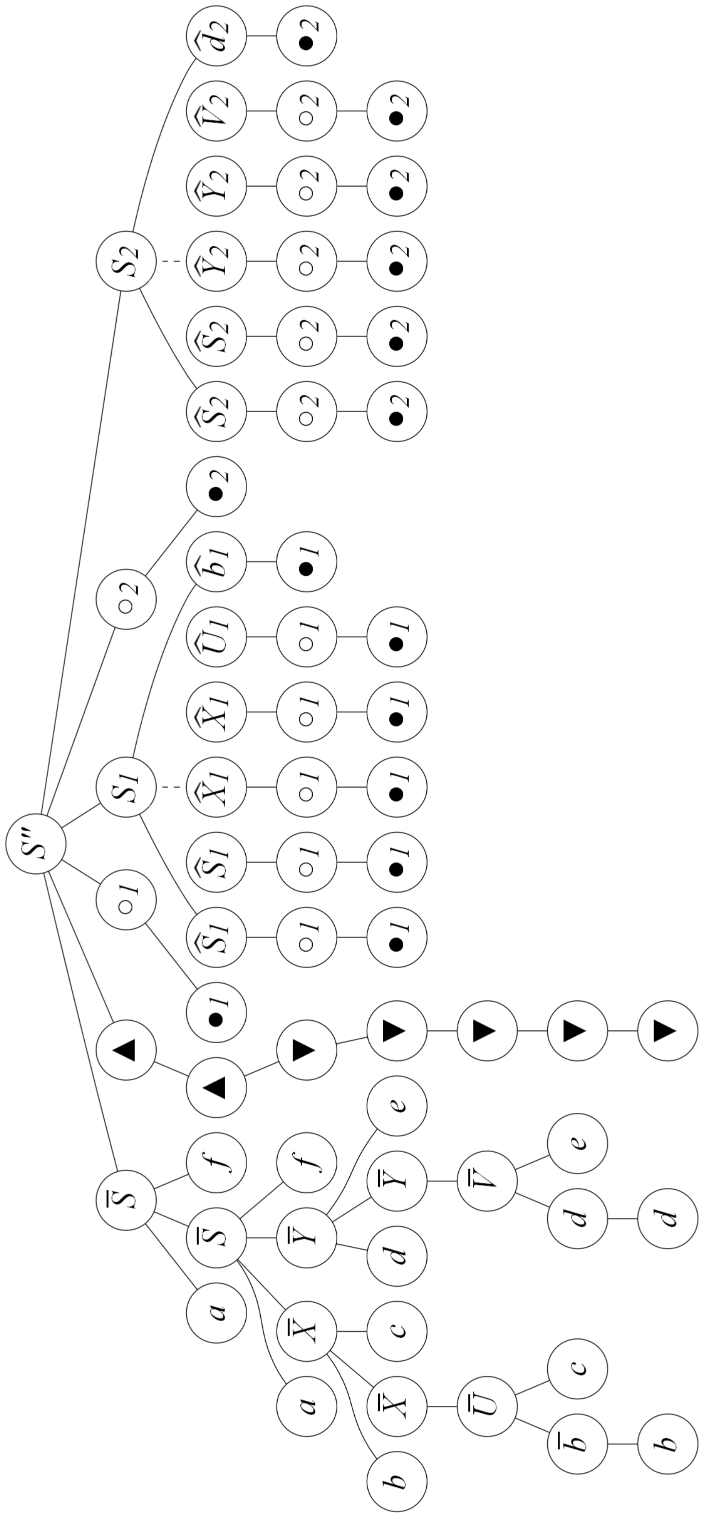

Figure 9.4: Illustration for the simulation of a tree controlled grammar (G, R) from Ex. 9.1 by propagating scattered context grammar G''. For the conciseness, the details of derivations $S_1' \Rightarrow^* \widehat{S_1}\widehat{S_1}\widehat{X_1}\widehat{X_1}\widehat{U_1}\widehat{b_1}$ and $S_2' \Rightarrow^* \widehat{S_2}\widehat{S_2}\widehat{Y_2}\widehat{Y_2}\widehat{V_2}\widehat{d_2}$ as well as the applications of $\blacktriangledown \rightarrow \blacklozenge$ and $\bullet_i \rightarrow \blacklozenge$ are omitted in this figure. The derivation tree of a sentence in ${}_{III\text{-}n\text{-}path}L(G, R)$ is given on the left-hand part of Fig. 9.2. For better readability, the figure is rotated.

Chapter 10

Summary

In this concluding chapter, we summarize the most interesting results achieved in this work and point out some important open questions. Based on the *State of the Art* in the area of restrictions placed upon the derivation trees summarized in Sec. 1.1 and Chap. II, this work deals in principle with three kinds of derivation-tree based restrictions, cut-based, path-based, and several-path-based, provided that each of these investigation areas are introduced and motivated in Chap. 7, Chap. 8, and Chap. 9, respectively. Note that each of the sections of Part III represents relatively independent derivation-tree-restriction-related topic, therefore each of them contains its own detailed concluding section (see Sec. 7.3.1, Sec. 8.3.2, Sec. 8.3.3, Sec. 8.3.4, Sec. 9.3.1, Sec. 9.3.2, Sec. 9.3.3, and Sec. 9.3.4). Next, we briefly summarize the results achieved in each of the three aforementioned derivation-tree-based restriction areas.

10.1 Cut Based Restriction

Concerning cut-based restriction placed upon the derivation trees, we have introduced two fundamental types of such kind of a restriction and thus, we have opened a new investigation area in derivation-tree-restricted models. Next, we have proved that both restrictions increase the generative power of context-free grammars so they characterize **RE** (see Th. 7.1 and Th. 7.2):

$$\textbf{ord-cut-TC}(\mathbf{CF}, \mathbf{REG}) = \textbf{cut-TC}(\mathbf{CF}, \mathbf{REG}) = \mathbf{RE}.$$

An important open problem consists of the investigation of cut controlled grammars where ε-productions are forbidden. Consequently, the grammars restricted in this way should be placed into the relation with some other well-known language families, such as **CS**, and the deciding the question whether or not:

$$\textbf{ord-cut-TC}_{\varepsilon}(\mathbf{CF}_{\varepsilon}, \mathbf{REG}) = \textbf{cut-TC}_{\varepsilon}(\mathbf{CF}_{\varepsilon}, \mathbf{REG}) = \mathbf{CS}.$$

Next open problem is the descriptional complexity of $\textbf{ord-cut-TC}(\mathbf{CF}, \mathbf{REG})$ and $\textbf{cut-TC}(\mathbf{CF}, \mathbf{REG})$. The results stated in Th. 7.1 and Th. 7.2 are based on the transformation of an unrestricted grammar in Pentonnen normal form. However, using Geffert normal form, the number of nonterminals in the resulting cut controlled grammar would be reduced.

Another future research idea is represented by the controlling the cuts of the derivation trees in which several types of subregular control languages are considered. In this way, the question whether or not a kind of a subregural language is enough to increase the generative power of controlled grammar properly. Consequently, the relation between the generative power of level-based and cut-based models restricted in this way would be founded out.

10.2 Path Based Restriction

As a continuation of the investigation of path-based restrictions introduced in [80] and studied in [81] and [82], we have considered the impact of ε-productions in path controlled grammars to the generative power and we have stated that ε-productions can be removed from a path controlled grammar without affecting its language (see Lem. 8.1 and Col. 8.2):

$$\textbf{path-TC}(\textbf{CF}, \textbf{CF}) = \textbf{path-TC}_\varepsilon(\textbf{CF}_\varepsilon, \textbf{CF}).$$

Next, we have established two Chomsky-like normal forms for path controlled grammars (see Def. 8.3 and Def. 8.4) and we have formulated algorithms (see Alg. 1 and Alg. 2) that transform a path controlled grammar in its normal form:

- Let $L \in \textbf{path-TC}(\textbf{CF}, \textbf{CF})$. Then, there exists a tree controlled grammar, (G, R), in 1st normal form such that $L = {}_{path}L(G, R)$.
- Let $L \in \textbf{path-TC}(\textbf{CF}, \textbf{CF})$. Then, there exists a tree controlled grammar, (G, R), in 2nd normal form such that $L = {}_{path}L(G, R)$.

A future investigation idea consists of the modifying a general parsing methods that are based on Chomsky normal form such that it will be able to parse path controlled grammars in a polynomial time.

Another practical motivated idea is represented the relation between path controlled grammars and the theory of pseudoknots. We have demonstrated several typical pseudoknots used in biology represented by the words (see Def. 8.5) of non-context-free languages. We have demonstrated some pseudoknots belong to **path-TC**(**LIN**, **LIN**) (see Th. 8.5, Th. 8.6, and Th. 8.7):

$$\{xyx^Ry^R : \ x, y \in \Sigma^* \text{ for some } \Sigma\} \in \textbf{path-TC}(\textbf{LIN}, \textbf{LIN}),$$
$$\{xyx^Rzz^Ry^R : \ x, y, z \in \Sigma^* \text{ for some } \Sigma\} \in \textbf{path-TC}(\textbf{LIN}, \textbf{LIN}),$$
$$\{xyx^Rzy^Rz^R : \ x, y, z \in \Sigma^* \text{ for some } \Sigma\} \in \textbf{path-TC}(\textbf{LIN}, \textbf{LIN}).$$

Apparently, there is a huge variety of another pseudoknot structures in biology. For example, $\{xyzx^Ry^Rz^R : \ x, y, z \in \Sigma^*\}$ and it is an open question whether or not those pseudoknots can be generated by tree controlled grammars with linear components that generate the language under path control.

The last presented result deals with a reflection on the generative power of path controlled grammars that has been considered as well-known (see [80]) for more than last ten years. However, we have presented an argument against the correctness of the proof given in [80] that states $\textbf{path-TC}(\textbf{CF}, \textbf{CF}) \subseteq \textbf{MAT}$. We have concluded the counterargument by stating that the aforementioned inclusion still may hold; however, it cannot be proved in the way given in [80]. More precisely, we have found the language tha can be generated by a grammar with controlled path. However, this language cannot be generated by the grammar obtained by the contruction introduced in [80]. Apparently, the generative power of path controlled grammar still represents an open problem.

10.3 Several Paths Based Restriction

It is well-know that path controlled grammars where the controlling grammar is regular characterize the same language class as its controlled grammar (see [80]) do. We have proved that the generative power of context-free grammars remains unchanged even if we restrict all paths in their derivation trees by regular languages (see Th. 9.1):

$$\mathbf{CF} = \mathbf{all\text{-}path\text{-}TC}(\mathbf{CF}, \mathbf{REG}).$$

We have introduced a generalization of path controlled grammars so that they generate the language under the restriction placed on not just one but several paths. Consequently, we have found some subsets of n-path controlled grammars so their languages satisfy pumping premises similar to well-known premises stated by pumping lemmata for **CF**, **LIN**, and **REG** (see Th. 9.2, Th. 9.3, an Th. 9.4)—more precisely:

- If $L \in \mathbf{I\text{-}n\text{-}path\text{-}TC}(\mathbf{CF}, \mathbf{LIN})$, $n \geq 1$, then there are two constants, $k, q \geq 0$, such that each word, $z \in L$, with $|z| \geq k$ can be written as
$$z = u_1 v_1 u_2 v_2 \dots u_{4n} v_{4n} u_{4n+1}$$
with $0 < |v_1 v_2 \dots v_{4n}| \leq q$ and for all $i \geq 1$, $u_1 v_1^i u_2 v_2^i \dots u_{4n} v_{4n}^i u_{4n+1} \in L$.

- If $L \in \mathbf{III\text{-}n\text{-}path\text{-}TC}(\mathbf{CF}, \mathbf{LIN})$, $n \geq 1$, then there are two constants, $k, q \geq 0$, such that each word, $z \in L$, with $|z| \geq k$ can be written as
$$z = u_1 v_1 u_2 v_2 \dots u_{2n+2} v_{2n+2} u_{2n+3}$$
with $0 < |v_1 v_2 \dots v_{2n+2}| \leq q$ and for all $i \geq 1$, $u_1 v_1^i u_2 v_2^i \dots u_{2n+2} v_{2n+2}^i u_{2n+3} \in L$.

- If $L \in \mathbf{V\text{-}n\text{-}path\text{-}TC}(\mathbf{CF}, \mathbf{LIN})$, $n \geq 1$, then there are two constants, $k, q \geq 0$, such that each word, $z \in L$, with $|z| \geq k$ can be written as
$$z = u_1 v_1 u_2 v_2 u_3 v_3 u_4 v_4 u_5$$
with $0 < |v_1 v_2 v_3 v_4| \leq q$ and for all $i \geq 1$, $u_1 v_1^i u_2 v_2^i u_3 v_3^i u_4 v_4^i u_5 \in L$.

A natural question that still remains open is whether or not there are similar pumping properties also for $\mathbf{II\text{-}n\text{-}path\text{-}TC}(\mathbf{CF}, \mathbf{LIN})$ and $\mathbf{IV\text{-}n\text{-}path\text{-}TC}(\mathbf{CF}, \mathbf{LIN})$.

We have also proved some closure properties (see Th. 9.5 and Th. 9.6), that is

- $\mathbf{n\text{-}path\text{-}TC}(T_G, T_R)$, i-$\mathbf{n\text{-}path\text{-}TC}(T_G, T_R)$ are closed under intersection with regular languages, union, and non-erasing homomorphism, for $T_G, T_R \in \{\mathbf{REG}, \mathbf{LIN}, \mathbf{CF}\}$ and $n \geq 1$, $i \in \{$**I, II, III, IV, V**$\}$;

- $\mathbf{I\text{-}n\text{-}path\text{-}TC}(\mathbf{CF}, \mathbf{LIN})$, $\mathbf{III\text{-}n\text{-}path\text{-}TC}(\mathbf{CF}, \mathbf{LIN})$, and $\mathbf{V\text{-}n\text{-}path\text{-}TC}(\mathbf{CF}, \mathbf{LIN})$ are not closed under concatenation, intersection, and complement, for $n \geq 1$.

Since n-path controlled grammars are a natural generalization of grammars with just one path controlled, we have studied several properties that are well-known for path controlled grammars in the case of controlling given n paths. Most importantly, we have tried to establish the generative power for n path controlled grammar. We have found the approximation of the generative power that can be applied also on grammars with just one path controlled. However, this approximation does not say too much since it is well-known that **PSC** is not closed under erasing homomorphism. Thus, we have informally concluded that: *„We either have the power to check what we need but not to remove it (using* **PSC***) or vice versa (using* **MAT***).“* More precisely, we have stated the following (see Th. 9.13):

Let $L \in$ **n-path-TC**(**CF**, **CF**), for $n \geq 1$. Then there exists $L' \in$ **PSC** with $L = h(L')$ for a homomorphism h.

Finally, we have studied several parsing properties of path-based restriction which is indisputably one of the most important language-class-characterizing property from the practical viewpoint. Formally, we have studied a polynomial time parsing possibilities and we have stated that (see Th. 9.15, Th. 9.16, and Th. 9.15):

- For a tree controlled grammar, (G, R) with an unambiguous context-free grammar, G, and a linear control language, R, the membership $x \in {}_{\textit{nc-n-path}}L(G, R)$, $n \geq 1$, is decidable in $O(|x|^k)$, for some $k \geq 0$.
- For a tree controlled grammar, (G, R), where G is a context-free grammar and $R \in$ **LIN**, there is a tree controlled grammar, (G', R'), such that G' does not contain unit productions and ${}_{\textit{nc-n-path}}L(G, R) = {}_{\textit{nc-n-path}}L(G', R')$, $n \geq 1$.
- For tree controlled grammar (G, R) where G is m-ambiguous LR grammar, $m \geq 1$, and an unambiguous language $R \in$ **LIN** , the membership $x \in {}_{\textit{nc-n-path}}L(G, R)$, $n \geq 1$, is decidable in $O(|x|^k)$, for some $k \geq 0$.

The significant disadvantage of n-path tree controlled grammars is that the number of n paths satisfying the properties of Def. 9.7 is strictly limited by the length of the right-hand sides of the productions of underlying context-free grammar. That is, given a general context-free grammar, G, and a linear language, R, controlling the paths, the membership of a certain language might be decidable. However, given the same context-free language as $L(G)$ as a context-free grammar, H, in Chomsky normal form together with R, we might not be able to find suitable path restriction to obtain the same language. On the other hand, the derivation trees of tree controlled grammars that generates their languages under n-path control by a linear language are constructed exactly as in context-free grammars and, in addition, we have to check some of their paths. Thus, there is actually great possibility to use well-known parsing methods for context-free languages to construct the derivation trees and to check their paths (see Sec. 9.3.4). However, in this viewpoint, n-path tree controlled grammars seems to be a quite fragile formalism since it requires a context-free grammar to have a production with at least n nonterminals on the right-hand side which ensures the division of n paths in a common node. Moreover, it means that any attempt to use a parsing method that transforms a context-free grammar into Chomsky normal form will basically destroy any path restriction with $n \geq 3$. Moreover, several nice properties of context-free grammars have been lost—e.g., decomposition based on pumping lemma for linear languages is potentially ambiguous and thus, the membership problem for i-**n-path-TC**, $i \in \{$**I, II, III, IV, V**$\}$, is potentially ambiguous also.

There are still many questions to be answered, namely generative power of grammars with path or paths controlled non-regularly, further closure properties, decision properties, etc. However, there are several other more general variants of path-based restriction. Indeed, a modification of the formalism such that the paths do not have to be divided in a common node of a derivation tree, or a variant where a path in a tree does not have to start in the root and end in a leaf of the tree.

Bibliography

[1] S. Abraham. Some questions of language theory. In *Proceedings of the 1965 conference on Computational linguistics*, 1965.

[2] S. Abraham. Some questions of phrase-structure grammars I. *Comput. Linguistics*, 4:10, 1965.

[3] A. V. Aho. Indexed grammars - an extension of context-free grammars. *Journal of the ACM*, 15:25, 1968.

[4] A. V. Aho, R. Sethi, and J. D. Ullman. *Compilers principles, techniques, and tools.* Addison-Wesley, 1986.

[5] A. V. Aho and J. D. Ullman. *The theory of parsing, translation, and compiling.* Prentice-Hall, Inc., 1972.

[6] J. W. Backus. The syntax and semantics of the proposed international algebraic language of the zurich ACM-GAMM conference. In *IFIP Congress*, 1959.

[7] Y. Bar-Hillel, M. Perles, and E. Shamir. On formal properties of simple phrase structure grammars. Technical report, Office of Naval Research, Information Systems Branch, 1960.

[8] I. Bellert. Relational phrase structure grammar and its tentative applications. *Information and Control*, 8:28, 1965.

[9] D. Billington. Using the context-free pumping lemma. *Journal of the ACM*, 36:21, 1993.

[10] N. Blum and R. Koch. Greibach normal form transformation revisited. *Information and Computation*, 150:7, 1999.

[11] A. Bondy. *Graph Theory.* Springer, 2010.

[12] R. V. Book. On the structure of context-sensitive grammars. *International Journal of Computer and Information Sciences*, 2:11, 1973.

[13] J. A. Brzozowski. A survey of regular expressions and their applications. *IRE Transactions on Electronic Computers*, EC-11:12, 1962.

[14] J.-Y. Cai and K. Samuthiram. A note on the pumping lemma for regular languages. Technical report, Department of Computer Science, SUNY Buffalo, 1996.

[15] M. Čermák. Basic properties of n-languages. In *Proceedings of the 17th Conference and Competition STUDENT EEICT 2011 Volume 3*, 2011.

[16] M. Čermák, J. Koutný, and A. Meduna. On n-language classes hierarchy. Acta Cybernetica, submitted.

[17] M. Čermák, J. Koutný, and A. Meduna. Parsing based on n-path tree-controlled grammars. *Theoretical and Applied Informatics*, 2011:16, 2012.

[18] M. Čermák and A. Meduna. n-accepting restricted pushdown automata systems. In *13th International Conference on Automata and Formal Languages*, 2011.

[19] N. Chomsky. Three models for the description of language. *IRE Transactions in Information Theory*, 1:12, 1956.

[20] N. Chomsky. On certain formal properties of grammars. *Information and Control*, 2:31, 1959.

[21] D. Colton. A restated pumping lemma for context-free languages. *ACM Special Interest Group on Automata and Computability Theory*, 24:1, 1993.

[22] A. B. Cremers. Normal forms for context-sensitive grammars. *Acta Informatica*, 3:15, 1973.

[23] E. Csuhaj-Varjú and G. Vaszil. Scattered context grammars generate any recursively enumerable language with two nonterminals. *Information Processing Letters*, 110:6, 2010.

[24] K. Čulik and H. A. Maurer. Tree controlled grammars. *Computing*, 19:11, 1977.

[25] J. Dassow, Gh. Păun, and A. Salomaa. Grammars with controlled derivations. In *Handbook of Formal Languages, Volume II*, 1997.

[26] J. Dassow and Gh. Păun. *Regulated Rewriting in Formal Language Theory.* Springer, 1989.

[27] J. Dassow, R. Stiebe, and B. Truthe. Two collapsing hierarchies of subregularly tree controlled languages. *Theoretical Computer Science*, 410:11, 2009.

[28] J. Dassow and B. Truthe. On two hierarchies of subregularly tree controlled languages. In *10th International Workshop on Descriptional Complexity of Formal Systems, DCFS 2008, Charlottetown, Prince Edward Island, Canada, July 16-18, 2008*, 2008.

[29] J. Dassow and B. Truthe. Subregularly tree controlled grammars and languages. In *Automata and Formal Languages - 12th International Conference AFL 2008, Balatonfured*, 2008.

[30] A. Ehrenfeucht and G. Rozenberg. An observation on scattered grammars. *Information Processing Letters*, 9:2, 1979.

[31] J. Eisner. Efficient normal-form parsing for combinatory categorial grammar. *Computing Research Repository*, cmp-lg/9605038:8, 1996.

[32] G. B. Enguix, M. D. Jiménez-López, and C. Martín-Vide, editors. *New Developments in Formal Languages and Applications.* Springer, 2008.

[33] Z. Ésik, C. Martín-Vide, and V. Mitrana, editors. *Recent Advances in Formal Languages and Applications.* Springer, 2006.

[34] P. A. Evans. Finding common RNA pseudoknot structures in polynomial time. *Journal of Discrete Algorithms*, 9:9, 2011.

[35] S. Ewert and A. van der Walt. A pumping lemma for random permitting context languages. *Theoretical Computer Science*, 270:8, 2002.

[36] B. Farwer, M. Jantzen, M. Kudlek, H. Rölke, and G. Zetzsche. Petri net controlled finite automata. *Fundamenta Informaticae*, 85:11, 2008.

[37] H. Fernau. Scattered context grammars with regulation. *Annals of the University of Bucharest. Mathematical Series.*, 45:9, 1996.

[38] H. Fernau and A. Meduna. A simultaneous reduction of several measures of descriptional complexity in scattered context grammars. *Information Processing Letters*, 86:6, 2003.

[39] R. W. Floyd. On ambiguity in phrase-structure languages. *Communications of the ACM*, 5:9, 1962.

[40] R. W. Floyd and R. Beigel. *The Language of Machines.* Freeman, 1994.

[41] I. Friš. Grammars with partial ordering of the rules. *Information and Control*, 12:11, 1968.

[42] A. Fujiyoshi. Analogical conception of chomsky normal form and greibach normal form for linear, monadic context-free tree grammars. *IEICE Transactions on Information and Systems*, e89-d:6, 2006.

[43] V. Geffert. Context-free-like forms for the phrase-structure grammars. *Lecture Notes in Computer Science*, 324:9, 1988.

[44] S. Ginsburg and H. G. Rice. Two families of languages related to ALGOL. *Journal of the ACM*, 9:22, 1962.

[45] S. Ginsburg and E. H. Spanier. Control sets on grammars. *Mathematical Systems Theory*, 2:19, 1968.

[46] S. Greibach and J. Hopcroft. Scattered context grammars. *Journal of Computer and System Sciences*, 3:15, 1969.

[47] A. (Annegret) Habel. Applications of the pumping lemma. *Lecture Notes in Computer Science*, 643:7, 1992.

[48] I. M. Havel. The theory of regular events ii. *Kybernetika*, 5:25, 1969.

[49] J. Higginbotham. English is not a context-free language. *Linguistic Inquiry*, 15:10, 1984.

[50] M. C. Ho. A survey of greibach normal form : transformation & analysis. Master's thesis, Hong Kong University of Science and Technology, 2006.

[51] J. E. Hopcroft and J. D. Ullman. *Introduction to Automata Theory.* Addison-Wesley, 1979.

[52] G. Horváth and B. Nagy. Pumping lemmas for linear and nonlinear context-free languages. *An International Scientific Journal of Sapientia University*, 2:16, 2010.

[53] F. WD Huang, L. YM Li, and Ch. M. Reidys. Sequence-structure relations of pseudoknot RNA. *Selected papers from the Seventh Asia-Pacific Bioinformatics Conference (APBC 2009)*, 1:19, 2009.

[54] C. W. Greider J. L. Chen. Functional analysis of the pseudoknot structure in human telomerase rna. In *Proceedings of the National Academy of Sciences of the United States of America*, 2005.

[55] J.A. Brzozowski. Derivates of Regular Expression. *Journal of the ACM*, 11:14, 1964.

[56] A. K. Joshi. Tree adjoining grammars: How much context-sensitivity is required to provide reasonable structural descriptions. Technical report, University of Pennsylvania, 1985.

[57] L. Kari and S. Seki. On pseudoknot-bordered words and their properties. *Journal of Computer and System Sciences*, 75:9, 2009.

[58] N. A. Khabbaz. A geometric hierarchy of languages. *Journal of Computer and System Sciences*, 8:19, 1974.

[59] S. C. Kleene. Representation of events in nerve nets and finite automata. In *Automata Studies.* Princeton University Press, 1956.

[60] J. Koutný. L-systems and their usage in practice. Bachelor thesis, Faculty of Information Technology, Brno University of Technology, 2006.

[61] J. Koutný. L-systems and their applications. Master's thesis, Faculty of Information Technology, Brno University of Technology, 2008.

[62] J. Koutný. L-systémy a jejich aplikace. In *Proceedings of the 14th Conference STUDENT EEICT 2008*, 2008.

[63] J. Koutný. Regular paths in derivation trees of context-free grammars. In *Proceedings of the 15th Conference STUDENT EEICT 2009 Volume 4*, 2009.

[64] J. Koutný. On n-path-controlled grammars. In *Proceedings of the 16th Conference STUDENT EEICT 2010 Volume 5*, 2010.

[65] J. Koutný. Syntax analysis of tree-controlled languages. In *Proceedings of the 17th Conference STUDENT EEICT 2011 Volume 3*, 2011.

[66] J. Koutný. On path-controlled grammars and pseudoknots. In *Proceedings of the 18th Conference STUDENT EEICT 2012 Volume 3*, 2012.

[67] J. Koutný, Z. Křivka, and A. Meduna. On grammars with controlled paths. Acta Cybernetica, submitted.

[68] J. Koutný, Z. Křivka, and A. Meduna. Pumping properties of path-restricted tree-controlled languages. In *7th Doctoral Workshop on Mathematical and Engineering Methods in Computer Science*, 2011.

[69] J. Koutný and A. Meduna. On normal forms and erasing rules in path controlled grammars. Schedae Informaticae, in press.

[70] J. Koutný and A. Meduna. Tree-controlled grammars with restrictions placed upon cuts and paths. *Kybernetika*, 48:11, 2012.

[71] M. Kudlek, C. Martin-Vide, A. Mateescu, and V. Mitrana. Contexts and the concept of mild context-sensitivity. *Linguistics and Philosophy*, 26:23, 2003.

[72] B. Leguy. Grammars witbout erasing rules. the OI case. In *Proceedings of the 6th Colloquium on Trees in Algebra and Programming (CAAP '81)*, 1981.

[73] M. K. Levitina. On some grammars with rules of global replacement. *Scientific-technical information (Nauchno-tehnichescaya informacii), Series 2*, page 5, 1972.

[74] A. Lindenmayer. Mathematical models for cellular interactions in development, I & II. *Journal of Theoretical Biology*, 18:36, 1968.

[75] A. Lindenmayer. Developmental systems without cellular interactions, their languages and grammars. *Journal of Theoretical Biology*, 30:30, 1971.

[76] A. Lindenmayer and G. Rozenberg. Developmental systems and languages. In *STOC: ACM Symposium on Theory of Computing (STOC)*, 1972.

[77] R. B. Lyngsø. Complexity of pseudoknot prediction in simple models. In *Automata, Languages and Programming: 31st International Colloquium, ICALP 2004, Turku, Finland, July 12-16, 2004. Proceedings*, 2004.

[78] R. B. Lyngsø and Ch.N. S. Pedersen. RNA pseudoknot prediction in energy-based models. *Journal of Computational Biology*, 7:19, 2000.

[79] G. Ma and Ch. M. Reidys. Canonical RNA pseudoknot structures. *Journal of Computational Biology*, 15:17, 2008.

[80] S. Marcus, C. Martín-Vide, V. Mitrana, and Gh. Păun. A new-old class of linguistically motivated regulated grammars. In *Walter Daelemans, Khalil Sima'an, Jorn Veenstra, Jakub Zavrel (Eds.): Computational Linguistics in the Netherlands 2000, Selected Papers from the Eleventh CLIN Meeting, Tilburg*, 2000.

[81] C. Martín-Vide and V. Mitrana. Further properties of path-controlled grammars. In *Proceedings of FG-MoL 2005: The 10th Conference on Formal Grammar and The 9th Meeting on Mathematics of Language*, 2005.

[82] C. Martin-Vide and V. Mitrana. Decision problems on path-controlled grammars. *International Journal of Foundations of Computer Science*, 18:11, 2007.

[83] T. Masopust. On the descriptional complexity of scattered context grammars. *Theoretical Computer Science*, 410:5, 2009.

[84] T. Masopust. Bounded number of parallel productions in scattered context grammars with three nonterminals. *Fundamenta Informaticae*, 99:8, 2010.

[85] A. Mateescu and A. Salomaa. Aspects of classical language theory. In *Handbook of formal languages, vol. 1*. Springer-Verlag New York, Inc., 1997.

[86] R. McNaughton. *Elementary computability, formal languages and automata.* Prentice-Hall, 1982.

[87] R. McNaughton and H. Yamada. Regular expressions and state graphs for automata. *IRE Transactions on Electronic Computers*, 9:9, 1960.

[88] A. Meduna. Context free derivations on word monoids. *Acta Informatica*, 27:6, 1990.

[89] A. Meduna. *Automata and Languages: Theory and Applications.* Springer Verlag, 2005.

[90] A. Meduna. *Elements of Compiler Design.* Taylor and Francis Informa plc, 2008.

[91] A. Meduna and D. Kolar. Regulated pushdown automata. *Acta Cybernetica*, 14:12, 2000.

[92] A. Meduna and J. Techet. *Scattered Context Grammars and their Applications.* WIT Press, 2010.

[93] A. Meduna and P. Zemek. *Regulated Grammars and Their Transformations.* Brno University of Technology, 2010.

[94] D. L. Milgram and A. Rosenfeld. A note on scattered context grammars. *Information Processing Letters*, 1:4, 1971.

[95] E. Moriya. Some remarks on state grammars and matrix grammars. *Information and Control*, 23:10, 1973.

[96] P. Naur et al. Report on the algorithmic language ALGOL 60. *Communications of the ACM*, 3:16, 1960.

[97] P. Navrátil. Parsing based on regulated grammars. Master's thesis, Faculty of Information Technology, Brno University of Technology, 2003.

[98] W. Ogden. A helpful result for proving inherent ambiguity. *Mathematical Systems Theory*, 2:4, 1968.

[99] M. A. Palis and S. M. Shende. Upper bounds on recognition of a hierarchy of non-context-free languages. *Theoretical Computer Science*, 98:28, 1992.

[100] M. A. Palis and S. M. Shende. Pumping lemmas for the control language hierarchy. *Mathematical Systems Theory*, 28:4, 1995.

[101] G. Păun. On the generative capacity of simple matrix grammars of finite index. *Information Processing Letters*, 7:3, 1978.

[102] G. Paun. On eliminating the lambda-rules from simple matrix grammars. *Fundamenta Informaticae*, 4:12, 1981.

[103] Gh. Păun. On the generative capacity of conditional grammars. *Information and Control*, 43:9, 1979.

[104] Gh. Păun. On the generative capacity of tree controlled grammars. *Computing*, 21:8, 1979.

[105] E. L. Post. A variant of a recursively unsolvable problem. *Bulletion of the American Mathematical Society*, 52:5, 1946.

[106] P. Prusinkiewicz and A. Lindenmayer. *The algorithmic beauty of plants.* Springer-Verlag, 1990.

[107] G. K. Pullum. On two recent attempts to show that english is not a CFL. *Computational Linguistics*, 10:5, 1984.

[108] G. K. Pullum and G. Gazdar. Natural languages and context-free languages. *Linguistics and Philosophy*, 4:34, 1982.

[109] Jr. R. L. Cannon. An algebraic technique for context-sensitive parsing. *International Journal of Computer and Information Sciences*, 5:20, 1976.

[110] S. Roman. *Latices and Ordered Sets.* Springer, 2009.

[111] D. J. Rosenkrantz. Programmed grammars and classes of formal languages. *Journal of the ACM*, 16:25, 1969.

[112] D. J. Rosenkrantz and R. E. Stearns. Properties of deterministic top down grammars. In *STOC: ACM Symposium on Theory of Computing (STOC)*, 1969.

[113] G. Rozenberg and A. Salomaa. *Handbook of Formal Languages (Vol 2) Linear Modeling: Background and Application.* Springer Verlag, 1997.

[114] A. Salomaa. Matrix grammars with a leftmost restriction. *Information and Control*, 20:7, 1972.

[115] A. K. Salomaa. *Formal Languages.* Academic Press, 1973.

[116] S. Shieber. Evidence against the context-freeness of natural language. *Linguistics and Philosophy*, 8:11, 1985.

[117] D. W. Staple and S. E. Butcher. Pseudoknots: RNA structures with diverse functions. *PLoS Biology*, 3:4, 2005.

[118] R. Stiebe. On the complexity of the control language in tree controlled grammars. In *Colloquium on the Occasion of the 50th Birthday of Victor Mitrana*, 2008.

[119] J. Techet. k-limited erasing performed by scattered context grammars. In *WFM*, 2007.

[120] J. W. Thatcher. Characterizing derivation trees of context-free grammars through a generalization of finite automata theory. *Journal of Computer and System Sciences*, 1:6, 1967.

[121] K. Thompson. Regular expression search algorithm. *Communications of the ACM*, 11:4, 1968.

[122] S. Turaev, J. Dassow, and M. H. Selamat. Language classes generated by tree controlled grammars with bounded nonterminal complexity. In *Descriptional Complexity of Formal Systems - 13th International Workshop, DCFS 2011, Gießen/Limburg, Germany, July 25-27, 2011. Proceedings*, 2011.

[123] S. Turaev, J. Dassow, and M. H. Selamat. Nonterminal complexity of tree controlled grammars. *Theoretical Computer Science*, 412:7, 2011.

[124] F. J. Urbanek. On Greibach normal form construction. *Theoretical Computer Science*, 40:3, 1985.

[125] A. P. J. van der Walt. Random context grammars. In *Proceedings of Symposium on Formal Languages*, 1970.

[126] G. Vaszil. On the descriptional complexity of some rewriting mechanisms regulated by context conditions. *Theoretical Computer Science*, 330:13, 2005.

[127] P. P. P. Velasco. *Matrix Graph Grammars.* VDM verlag, 2008.

[128] P. P. P. Velasco. Matrix graph grammars: Transformation of restrictions. *Computing Research Repository*, abs/0912.2160:25, 2009.

[129] D. Wätjen. Regulation of k-limited ET0L systems. *International Journal of Computational Methods*, 47:13, 1993.

[130] B. Wegbreit. A generator of context-sensitive languages. *Journal of Computer and System Sciences*, 3:6, 1969.

[131] D. J. Weir. A geometric hierarchy beyond context-free languages. *Theoretical Computer Science*, 104:7, 1992.

[132] D. S. Wise. A strong pumping lemma for context-free languages. *Theoretical Computer Science*, 3:11, 1976.

[133] D. Wood. *Theory of computation.* Wiley, 1987.

[134] W. A. Woods. Context-sensitive parsing. *Communications of the ACM*, 13:9, 1970.

[135] A. Yehudai. A note on the pumping lemma for regular languages. *Information Processing Letters*, 9:2, 1979.

[136] D. H. Younger. Recognition and parsing of context-free languages in time n^3. *Information and Control*, 10:20, 1967.

[137] S. Yu. A pumping lemma for deterministic context-free languages. *Information Processing Letters*, 31:5, 1989.

[138] P. Zemek. On erasing rules in regulated grammars. Master's thesis, Faculty of Information Technology, Brno University of Technology, 2010.

[139] G. Zetzsche. Toward understanding the generative capacity of erasing rules in matrix grammars. *International Journal of Foundations of Computer Science*, 22:16, 2011.