Tomáš Masopust

Formal Models: Regulation and Reduction

Dissertation thesis

Faculty of Information Technology
Brno University of Technology
Brno, Czech Republic

Dissertation thesis

Cover design 2007 by Dagmar Hejduková

Published by Faculty of Information Technology,
Brno University of Technology, Brno, Czech Republic

Printed by MJ servis, spol. s r.o.

ISBN 978-80-214-3550-6

Preface

The subject of this monograph is divided into two parts—regulated and reduced formal models.

The first part introduces and studies *self-regulating finite and pushdown automata*. In essence, these automata regulate the use of their rules by a sequence of rules applied during the previous moves. A special attention is paid to *turns* defined as moves during which a self-regulating automaton starts a new self-regulating sequence of moves.

Based on the number of turns, two infinite hierarchies of language families resulting from two variants of these automata are established (see Sections 4.1.1 and 4.1.2). Section 4.1.1 demonstrates that in case of self-regulating finite automata these hierarchies coincide with the hierarchies resulting from parallel right linear and right linear simple matrix grammars, so the self-regulating finite automata can be viewed as the automaton counterparts to these grammars. Finally, both infinite hierarchies are compared. In addition, Section 4.1.2 discusses some results and open problems concerning self-regulating pushdown automata.

The second part studies descriptional complexity of partially parallel grammars (Section 5.1) and grammars regulated by context conditions (Section 5.2) with respect to the number of nonterminals and a special type of productions.

Specifically, Chapter 5 proves that every recursively enumerable language is generated (i) by a scattered context grammar with no more than four non-context-free productions and four nonterminals, (ii) by a multisequential grammar with no more than two selectors and two nonterminals, (iii) by a multicontinuous grammar with no more than two selectors and three nonterminals, (iv) by a context-conditional grammar of degree $(2,1)$ with no more than six conditional productions and seven nonterminals, (v) by a simple context-conditional grammar of degree $(2,1)$ with no more than seven conditional productions and eight nonterminals, (vi) by a generalized forbidding grammar of degree two and index six with no more than ten conditional productions and nine nonterminals, (vii) by a generalized forbidding grammar of degree two and index four with no more than eleven conditional productions and ten nonterminals, (viii) by a generalized forbidding grammar of degree two and index nine with no more than eight conditional productions and ten nonterminals, (ix) by a generalized forbidding grammar of degree two and unlimited index with no more than nine conditional productions and eight nonterminals, (x) by a semi-conditional grammar of degree $(2, 1)$ with no more than seven conditional productions and eight nonterminals, and (xi) by a simple semi-conditional grammar of degree $(2, 1)$ with no more than nine conditional productions and ten nonterminals.

Chapter 2 contains basic definitions and the notation used during this monograph. Chapter 3 then summarizes the previous known results related to the studied formal models; regulated automata and descriptional complexity of partially parallel grammars and grammars regulated by context conditions. Chapter 4 studies self-regulating automata, and Chapter 5 presents the new results concerning descriptional complexity of partially parallel grammars and grammars regulated by context conditions.

Acknowledgment

This work was supported by the Czech Ministry of Education under the Research Plan No. MSM 0021630528 (Security-Oriented Research in Information Technology), FRVŠ grant No. FR762/2007/G1 (Teaching of Formal Language Theory at the International Level of EU), and the Czech Grant Agency project No. 102/05/H050 (Integrated approach to education of PhD students in the area of parallel and distributed systems).

The author would like to thank prof. Václav Snášel and doc. Dušan Kolář for their careful reading of the previous version of this monograph. However, all mistakes appearing in the final version of this monograph are only his faults.

The author thanks his teacher and friend prof. Alexander Meduna.

Contents

1 Introduction ... 1

2 Notation and Basic Definitions ... 5
2.1 Alphabets and Strings ... 5
2.2 Languages and Language Operations ... 6
2.3 Grammars ... 8
2.4 Automata ... 10
2.4.1 Finite Automata ... 11
2.4.2 Pushdown Automata ... 12

3 Current Concepts and Results ... 13
3.1 Regulated Formal Systems ... 13
3.2 Descriptional Complexity of Grammars ... 15

Part I Self-Regulating Automata

4 Self-Regulating Automata ... 19
4.1 Definitions and Examples ... 19
4.1.1 Self-Regulating Finite Automata ... 20
4.1.2 Self-Regulating Pushdown Automata ... 22
4.2 Self-Regulating Finite Automata ... 23
4.2.1 First-Move Self-Regulating Finite Automata ... 23
4.2.2 All-Move Self-Regulating Finite Automata ... 28
4.2.3 Language Families Accepted by n-first-SFAs and n-all-SFAs 33
4.3 Self-Regulating Pushdown Automata ... 34
4.3.1 All-Move Self-Regulating Pushdown Automata ... 35
4.3.2 First-Move Self-Regulating Pushdown Automata ... 36
4.3.3 Open Problems ... 37

Part II Descriptional Complexity

5 Descriptional Complexity 41
5.1 Partially Parallel Grammars 42
5.1.1 Scattered Context Grammars 42
5.1.2 Propagating Scattered Context Grammars 47
5.1.3 Multisequential Grammars 53
5.1.4 Multicontinuous Grammars 55
5.2 Context-Conditional Grammars 57
5.2.1 Context-Conditional Grammars with Linear and Regular Productions 60
5.2.2 Simple Context-Conditional Grammars 62
5.2.3 Generalized Forbidding Grammars 63
5.2.4 Generalized Permitting Grammars 75
5.2.5 Semi-Conditional Grammars 75
5.2.6 Simple Semi-Conditional Grammars 78

6 Conclusion 83

Index 87

References 89

1

Introduction

At the end of 50’s, linguist Naom Chomsky introduced the well-known hierarchy of languages (regular, context-free, context-sensitive, and recursively enumerable languages), which is in his honour called Chomsky hierarchy. His work inspired mathematicians and theoretical computer scientists, who gave that theory the needed formal shape convenient for its application in informatics. Thereby, formal language theory was established.

In classical formal language theory, there are three main approaches to formal languages.

1. Grammatical approach—language generation.
2. Automata approach—language recognition.
3. Algebraic approach—based on algebraic properties of languages and families of languages, such as closure properties under some language operations, etc. (see [Gin75]).

According to the previous approaches, this monograph is divided into two parts.

The first part, consisting of Chapter 4, is concerning the automata approach to the theory of formal languages. This chapter introduces and studies so-called self-regulating automata (see [MM07d]). Automata theory has, over its history, modified and restricted classical automata in many ways (see [Cou77, FR68, GGH67, GS68, Gre69, Med03, Med06, Sak81, Sir71, Val89]). Recently, regulated automata have been introduced and studied in [MK00, MK02]. In essence, these automata regulate the use of their rules according to which they make moves by control languages. This monograph continues with this topic by defining and investigating *self-regulating finite (pushdown) automata*. Instead of prescribed control languages, the self-regulating automata restrict the selection of a rule according to which the current move is made by a rule according to which a previous move was made.

To give a more precise insight into self-regulating automata, consider a finite automaton, M, with a finite binary relation, R, over M’s rules. Furthermore, suppose that M makes a sequence of moves, ρ, that leads to the acceptance of a string, so ρ can be expressed as a concatenation of $n+1$ consecutive subsequences, $\rho = \rho_0\rho_1\dots\rho_n$, $|\rho_i| = |\rho_j|$, $0 \le i, j \le n$, in which r_i^j denotes the rule according to which the ith move in ρ_j is made, for all $0 \leq j \leq n$ and $1 \leq i \leq |\rho_j|$ (as usual, $|\rho_j|$ denotes the length of ρ_j).

If for all $0 \leq j < n$, $(r_1^j, r_1^{j+1}) \in R$, then M represents an *n-turn first-move self-regulating finite automaton with respect to R*.

If for all $0 \leq j < n$ and, in addition, for all $1 \leq i \leq |\rho_i|$, $(r_i^j, r_i^{j+1}) \in R$, then M represents an *n-turn all-move self-regulating finite automaton with respect to R*.

Based on the number of turns, two infinite hierarchies of language families that lie between the families of regular and context-sensitive languages are established. First, a demonstration that n-turn first-move self-regulating finite automata give rise to an infinite hierarchy of language families coinciding with the hierarchy resulting from $(n+1)$-parallel right linear grammars (see [RW73, RW75, Woo73, Woo75]) is given. Recall that n-parallel right linear grammars generate a proper language subfamily of the language family generated by $(n+1)$-parallel right linear grammars (see Theorem 5 in [RW75]). As a result, n-turn first-move self-regulating finite automata accept a proper language subfamily of the language family accepted by $(n+1)$-turn first-move self-regulating finite automata, for all $n \geq 0$. Similarly, a proof that n-turn all-move self-regulating finite automata give rise to an infinite hierarchy of language families coinciding with the hierarchy resulting from $(n+1)$-right linear simple matrix grammars (see [DP89, Iba70, Woo75]) is given. As n-right linear simple matrix grammars generate a proper subfamily of the language family generated by $(n+1)$-right linear simple matrix grammars (see Theorem 1.5.4 in [DP89]), n-turn all-move self-regulating finite automata accept a proper language subfamily of the language family accepted by $(n+1)$-turn all-move self-regulating finite automata. Furthermore, since the families of right linear simple matrix languages coincide with the language families accepted by multitape nonwriting automata (see [FR68]) and by finite-turn checking automata (see [Sir71]), the all-move self-regulating finite automata characterize these families, too. Finally, the results about both infinite hierarchies are summarized.

Next, *self-regulating pushdown automata* are discussed. Regarding all-move self-regulating pushdown automata, a proof that all-move self-regulating pushdown automata do not give rise to any infinite hierarchy analogical to hierarchies resulting from the self-regulating finite automata is given. It is shown that while zero-turn all-move self-regulating pushdown automata define the family of context-free languages, one-turn all-move self-regulating pushdown automata define the family of recursively enumerable languages.

On the other hand, as far as first-move self-regulating pushdown automata are concerned, it is an easy observation that zero-turn first-move self-regulating pushdown automata define the family of context-free languages. However, the question whether these automata define an infinite hierarchy with respect to the number of turns or not is open.

The second part of this monograph, consisting of Chapter 5, is concerning the grammatical approach. Specifically, it studies descriptional complexity of partially parallel grammars and grammars regulated by context conditions. The main aim of descriptional complexity of grammars is to describe grammars in a reduced and succinct way (see pages 145–148 of Volume 2 in [RS97] for an overview). This trend of formal language theory has recently so intensified that an annual international conference *Descriptional Complexity of Formal Systems* is held to discuss this specific topic (see [MPPW05, LRCP06, GP07] for its latest proceedings). As a central topic, this investigation of descriptional complexity studies how to reduce the number of grammatical components, such as the number of nonterminals or (special) productions.

Consider a family of languages, $\mathscr{L}$, and a family of grammars, $\mathscr{G}$, such that $L \in \mathscr{L}$ if and only if there is a grammar $G \in \mathscr{G}$ such that $L = L(G)$. To reduce the number of nonterminals means to find a natural number (if it exists), k, such that for every language $L \in \mathscr{L}$, there is a grammar $G \in \mathscr{G}$ such that the set of all G's nonterminals, N, contains no more than k elements, $|N| \leq k$, and G generates L, $L = L(G)$. In other words, the question is what is the minimal k such that there is a subfamily, $\mathscr{H}$, of $\mathscr{G}$ consisting of grammars having no more than k nonterminals such that any language from $\mathscr{L}$ is generated by a grammar from $\mathscr{H}$. The reduction of special productions is defined analogously, i.e., the aim is to find a natural number (if it exists), l, such that for every language $L \in \mathscr{L}$, there is a grammar $G \in \mathscr{G}$ with P being the set of all its productions, $P = P' \cup P''$, where P'' is the set of all special productions, such that $|P''| \leq l$ and $L = L(G)$. For instance, let P' be the set of all context-free and P'' the set of all remaining productions of P.

This monograph studies the simultaneous reduction of both the number of nonterminals and the number of special productions. In other words, in case of studied grammars, it is well-known that there are natural numbers k and l such that there is a subfamily, $\mathscr{H}$, of $\mathscr{G}$ having no more than k nonterminals and l special productions such that any language from $\mathscr{L}$ is generated by a grammar from $\mathscr{H}$. We decrease these numbers as follows.

The first section of Chapter 5 studies descriptional complexity of scattered context, multisequential, and multicontinuous grammars (see [DP89, KR83a, Med97a, Med97c, Med98a, Med00b, Med00c, Med02, MF03a, MF03b, Vas05] for more details). These grammars are ordinary context-free grammars, where a limited number of productions is allowed to be parallelly applied in one derivation step. Recall that every recursively enumerable language was shown to be generated

(1) by a scattered context grammar with no more than five nonterminal symbols and two non-context-free productions (see [Vas05]);
(2) by a multisequential grammar with no more than six nonterminal symbols (see [Med97c]); and
(3) by a multicontinuous grammar with no more than six nonterminal symbols (see [Med98a]).

In this monograph, these results are improved (see [MMa]). Specifically, it proves that every recursively enumerable language is generated

(A) by a scattered context grammar with no more than four nonterminal symbols and four non-context-free productions;
(B) by a multisequential grammar with no more than two nonterminal symbols and two selectors; and

(C) by a multicontinuous grammar with no more than three nonterminal symbols and two selectors.

The second section of Chapter 5 studies descriptional complexity of context-conditional grammars. Context-conditional grammars are context-free grammars in which two sets of strings, called a permitting and a forbidding context, are attached to each production. Such a production is then applicable if each element of its permitting context occurs in the current sentential form while none of its forbidding context does.

Many variants of these grammars that differ in requirements put on their permitting and forbidding contexts are studied in the literature, such as generalized forbidding, semi-conditional, or simple semi-conditional grammars (see [DP89, Kel84, MŠ02, MŠ05, Pău85]). All these grammars are proved to be able to generate the family of recursively enumerable languages. Specifically, recall that every recursively enumerable language was shown to be generated

(1) by a context-conditional grammar of degree $(1,1)$ (however, the number of conditional productions and nonterminals is not limited, see [DP89, May72, Sal73]);
(2) by a generalized forbidding grammar of degree two with no more than thirteen conditional productions and fifteen nonterminals (see [MŠ03]); and
(3) by a simple semi-conditional grammar of degree $(2,1)$ with no more than ten conditional productions and twelve nonterminals (see [Vas05]).

This monograph improves these results (see [Mas06, Mas07b, MM07a, MM07c, MM07b]). Specifically, it proves that every recursively enumerable language is generated

(A) by a context-conditional grammar of degree $(2,1)$ with no more than seven conditional productions and eight nonterminals;
(B) by a generalized forbidding grammar of degree two and index six with no more than ten conditional productions and nine nonterminals;
(C) by a generalized forbidding grammar of degree two and index four with no more than eleven conditional productions and ten nonterminals;
(D) by a generalized forbidding grammar of degree two and index nine with no more than eight conditional productions and ten nonterminals;
(E) by a generalized forbidding grammar of degree two and unlimited index with no more than nine conditional productions and eight nonterminals;
(F) by a simple semi-conditional grammar of degree $(2,1)$ with no more than nine conditional productions and ten nonterminals; and
(G) by a semi-conditional grammar of degree $(2,1)$ with no more than seven conditional productions and eight nonterminals.

In fact, except for result (E), all these results are established for grammars with context conditions represented by strings consisting solely of nonterminals as opposed to the previous results that allow terminals to appear in them as well.

2

Notation and Basic Definitions

The set of all natural numbers is denoted by $\mathbb{N}$. The set of all natural numbers with zero is denoted by $\mathbb{N}_0$. The cardinality of a set, A, is denoted by $|A|$. For two sets, A and B, $A \subseteq B$ denotes that A is a subset of B; $A \subset B$ denotes that $A \subseteq B$ and $A \neq B$, i.e. A is a proper subset of B.

2.1 Alphabets and Strings

An *alphabet* is an arbitrary finite nonempty set of elements, which are called *symbols*. A finite sequence, w, of symbols forms a *string*. The *empty string*, denoted by ε, is the string that contains no symbols. The length of w, $|w|$, is the number of all symbols in w.

Let x and y be two strings over an alphabet T. Then, xy is the *concatenation* of x and y. The following equation is an immediate consequence of the definition;

$$x\varepsilon = \varepsilon x = x\,.$$

Definition 2.1.1. Let x be a string over an alphabet T. For $i \in \mathbb{N}_0$, the ith *power* of x is defined as

1. $x^0 = \varepsilon$
2. $x^i = xx^{i-1}$

Observe that for any string x,

$$x^i x^j = x^j x^i = x^{i+j}\,,$$

for any $i, j \in \mathbb{N}_0$.

Definition 2.1.2. Let x be a string over an alphabet T. The *reversal* of x, x^R, is defined as

1. $\varepsilon^R = \varepsilon$

2. if $x = a_1 \ldots a_n$, for some $n \in \mathbb{N}$, and $a_i \in \Sigma$, for $i = 1, \ldots, n$, then $(a_1 \ldots a_n)^R = a_n \ldots a_1$.

Definition 2.1.3. Let x and y be two strings over an alphabet T. Then, x is a *substring* of y if there exist two strings z and z' over T so that $zxz' = y$. If $z = \varepsilon$, then x is a *prefix* of y. If $z' = \varepsilon$, then x is a *suffix* of y. Moreover, if $x \notin \{\varepsilon, y\}$, then x is a *proper* substring (prefix, suffix) of y.

2.2 Languages and Language Operations

Let T be an alphabet and let T^* denote the set of all strings over T. Set $T^+ = T^* - \{\varepsilon\}$. In other words, T^+ denotes the set of all nonempty strings over T.

A *language*, L, over T is a subset of T^*, i.e.

$$L \subseteq T^* .$$

(Sometimes, if it does not lead to confusion, a singleton set $\{a\}$ is denoted as a.)

As languages are sets, the common set operations can be applied to them (such as union, intersection, difference, and complement). That is, for two languages L_1 and L_2,

$$\begin{aligned} L_1 \cup L_2 &= \{x : x \in L_1 \text{ or } x \in L_2\}, \\ L_1 \cap L_2 &= \{x : x \in L_1 \text{ and } x \in L_2\}, \\ L_1 - L_2 &= \{x : x \in L_1 \text{ and } x \notin L_2\}. \end{aligned}$$

Consider a language, L, over an alphabet T. The *complement* of L, $\bar{L}$, is defined as

$$\bar{L} = T^* - L .$$

A language, L, is said to be *finite* if $|L| = n$, for some $n \in \mathbb{N}_0$; otherwise, L is said to be *infinite*.

The basic language operations follow.

Definition 2.2.1. 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 2.2.2. Let L be a language. The *reversal* of L, L^R, is defined as

$$L^R = \{x^R : x \in L\} .$$

Definition 2.2.3. Let L be a language. For $i \in \mathbb{N}_0$, the ith *power* of L, L^i, is defined as

1. $L^0 = \varepsilon$
2. $L^i = LL^{i-1}$

Definition 2.2.4. Let L be a language. The *Kleene closure* of L, L^*, is defined as

$$L^* = \bigcup_{i=0}^{\infty} L^i .$$

Definition 2.2.5. Let L be a language. The *positive closure* of L, L^+, is defined as

$$L^+ = \bigcup_{i=1}^{\infty} L^i .$$

Definition 2.2.6. Let $f : T^* \to 2^{U^*}$ be a mapping, T, U alphabets. If f satisfies the following conditions, then f is said to be a *substitution*.

1. $f(\varepsilon) = \{\varepsilon\}$,
2. $f(xy) = f(x)f(y)$, where $x, y \in T^*$.

f is said to be *finite* if $f(a)$ is a finite language, for all $a \in T$. For any language $L \subseteq T^*$,

$$f(L) = \bigcup_{x \in L} f(x) .$$

The substitution f is called *nonerasing* if $\varepsilon \notin f(a)$, for any $a \in T$.

A *homomorphism* is a substitution f such that $|f(a)| = 1$, for all $a \in T$. Let f be a homomorphism. Then, the *inverse homomorphic image* of L is the set

$$f^{-1}(L) = \{x \in T^* : f(x) \in L\} ,$$

and, for strings,

$$f^{-1}(w) = \{x \in T^* : f(x) = w\} .$$

Definition 2.2.7. A *right quotient* of a language L with a language K is the set

$$L/K = \{w : wx \in L, \text{ for some } x \in K\} .$$

Definition 2.2.8. Let $\mathscr{F}$ be a family of languages and $\mathscr{O}$ be an n-ary language operation. $\mathscr{F}$ is *closed* under the operation $\mathscr{O}$ if, for any languages $L_1, \ldots, L_n \in \mathscr{F}$, $\mathscr{O}(L_1, \ldots, L_n) \in \mathscr{F}$.

Definition 2.2.9. Let w be a string over an alphabet T. Then,

$$sub(w) = \{u : u \text{ is a substring of } w\} ,$$

and

$$alph(w) = \{a \in T : a \text{ appears in } w\} .$$

For any language, L, over T,

$$alph(L) = \bigcup_{w \in L} alph(w) .$$

Definition 2.2.10. For a finite subset $W \subseteq T^*$, T is an alphabet, $\max(W)$ is the minimal nonnegative integer n such that $|x| \leq n$, for all $x \in W$.

Definition 2.2.11. For integers $n_1, \ldots, n_k$, $k \in \mathbb{N}$, $\max\{n_1, \ldots, n_k\}$ denotes the maximum of $n_1, \ldots, n_k$.

2.3 Grammars

In this section, devices generating languages are defined. Such devices are called grammars and play the main role in formal language theory.

Definition 2.3.1. A *grammar*, G, is a quadruple

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

where

- N is a *nonterminal* alphabet,
- T is a *terminal* alphabet such that $N \cap T = \emptyset$,
- P is a finite set of *productions* of the form

  $$u \to v,$$

  where $u \in V^*NV^*$ and $v \in V^*$; V denotes the *total* alphabet of G, i.e. $V = N \cup T$.
- $S \in N$ is the *start* symbol.

Every grammar $G = (N, T, P, S)$ defines a binary relation of *direct derivation* on the set V^* denoted by $\Rightarrow$ and defined as

$$x \Rightarrow y$$

provided that

1. there is a production $u \to v \in P$ and
2. strings $x_1, x_2 \in V^*$ such that
   - $x = x_1 u x_2$ and
   - $y = x_1 v x_2$.

If $x, y \in V^*$ and $m \in \mathbb{N}$, then $x \Rightarrow^m y$ if and only if there is a sequence $x_0 \Rightarrow x_1 \Rightarrow \ldots \Rightarrow x_m$, where $x_0 = x$ and $x_m = y$. We write $x \Rightarrow^+ y$ if and only if there is $m \in \mathbb{N}$ such that $x \Rightarrow^m y$, and $x \Rightarrow^* y$ if and only if $x = y$ or $x \Rightarrow^+ y$. In other words, $\Rightarrow^+$ and $\Rightarrow^*$ are the transitive and the reflexive and transitive closures of $\Rightarrow$, respectively.

The elements of V^* that can be derived from the start symbol, S, are called *sentential forms* of $G = (N, T, P, S)$. More precisely, $x \in V^*$ is a sentential form if

$$S \Rightarrow^* x.$$

If x does not contain nonterminals, then x is called a *sentence*. If x is a sentence, then $S \Rightarrow^* x$ is said to be a *terminal derivation*. The set of all sentences is the language *generated* by G, denoted by $L(G)$, i.e.

$$L(G) = \{w \in T^* : S \Rightarrow^* w\}.$$

Grammars G_1 and G_2 are said to be *equivalent* if and only if they generate the same language, i.e.

$$L(G_1) = L(G_2).$$

Chomsky Hierarchy of Languages

At the end of 50's, linguist Naom Chomsky separated grammars into four basic groups according to limitations put on their productions. Chomsky hierarchy distinguishes the following four basic types of grammars:

type 0: Any grammar is a type 0 grammar.
type 1: A grammar is a type 1 (or context-sensitive) grammar if all its productions are of the form $u \to v$ with $|u| \leq |v|$; except for the case $S \to \varepsilon$, where S does not occur on the right-hand side of any production.
type 2: A grammar is a type 2 (or context-free) grammar if all its productions are of the form $u \to v$ with $u \in N$.
type 3: A grammar is a type 3 (or regular) grammar if all its productions are of the form $u \to v$ with $u \in N$ and $v \in TN \cup T \cup \{\varepsilon\}$.

The hierarchy of grammars establishes the hierarchy of languages. A language, L, is said to be regular (context-free, context-sensitive, recursively enumerable) if there is a regular (context-free, context-sensitive, type 0) grammar, G, such that $L = L(G)$. These families of languages are denoted by REG, CF, CS, and RE, respectively. The following theorem holds (see [Med00a]).

Theorem 2.3.2. $REG \subset CF \subset CS \subset RE$.

Definition 2.3.3. Let $G = (N,T,P,S)$ be a grammar. G is in the *Kuroda normal form* if each production in P is in one of the following four forms

1. $AB \to CD$,
2. $A \to BC$,
3. $A \to a$,
4. $A \to \varepsilon$,

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

In addition, if for each production of the form $AB \to CD$ we have $A = C$, then G is in the *Penttonen normal form*.

Proofs of the following theorem can be found in [Med00a, Pen74].

Theorem 2.3.4. *Let L be a recursively enumerable language. Then, there is a grammar G in the Kuroda (Penttonen) normal form such that $L = L(G)$.*

The following three normal forms are fundamental for the results concerning descriptional complexity of grammars proved in this monograph.

Definition 2.3.5. Let $G = (N,T,P,S)$ be a grammar.

1. G is in the *first Geffert normal form* if it is of the form

$$G = (\{S,A,B,C\},T,P \cup \{ABC \to \varepsilon\},S),$$

where P contains context-free productions of the form

$$\begin{array}{ll} S \to uSa, & \text{where } u \in \{A,AB\}^*,\ a \in T, \\ S \to uSv, & \text{where } u \in \{A,AB\}^*,\ v \in \{BC,C\}^*, \\ S \to uv, & \text{where } u \in \{A,AB\}^*,\ v \in \{BC,C\}^*. \end{array}$$

2. G is in the *second Geffert normal form* if it is of the form

$$G = (\{S,A,B,C,D\},T,P \cup \{AB \to \varepsilon, CD \to \varepsilon\},S),$$

where P contains context-free productions of the form

$$\begin{array}{ll} S \to uSa, & \text{where } u \in \{A,C\}^*,\ a \in T, \\ S \to uSv, & \text{where } u \in \{A,C\}^*,\ v \in \{B,D\}^*, \\ S \to uv, & \text{where } u \in \{A,C\}^*,\ v \in \{B,D\}^*. \end{array}$$

3. G is in the *third Geffert normal form* if it is of the form

$$G = (\{S,A,B\},T,P \cup \{ABBBA \to \varepsilon\},S),$$

where P contains context-free productions of the form

$$\begin{array}{ll} S \to uSa, & \text{where } u \in \{AB,ABB\}^*,\ a \in T, \\ S \to uSv, & \text{where } u \in \{AB,ABB\}^*,\ v \in \{BA,BBA\}^*, \\ S \to uv, & \text{where } u \in \{AB,ABB\}^*,\ v \in \{BA,BBA\}^*. \end{array}$$

The following three theorems are proved in [Gef88a, Gef91b].

Theorem 2.3.6. *Let L be a recursively enumerable language, then there is a grammar, G, in the first Geffert normal form such that $L = L(G)$.*

In addition, any terminal derivation in G is of the form $S \Rightarrow^* w_1w_2w$ by productions from P, where $w_1 \in \{A,AB\}^*$, $w_2 \in \{BC,C\}^*$, $w \in T^*$, and $w_1w_2w \Rightarrow^* w$ is derived by $ABC \to \varepsilon$.

Theorem 2.3.7. *Let L be a recursively enumerable language, then there is a grammar, G, in the second Geffert normal form such that $L = L(G)$.*

In addition, any terminal derivation in G is of the form $S \Rightarrow^* w_1w_2w$ by productions from P, where $w_1 \in \{A,C\}^*$, $w_2 \in \{B,D\}^*$, $w \in T^*$, and $w_1w_2w \Rightarrow^* w$ is derived by $AB \to \varepsilon$ and $CD \to \varepsilon$.

Theorem 2.3.8. *Let L be a recursively enumerable language, then there is a grammar, G, in the third Geffert normal form such that $L = L(G)$.*

In addition, any terminal derivation in G is of the form $S \Rightarrow^* w_1w_2w$ by productions from P, where $w_1 \in \{AB,ABB\}^*$, $w_2 \in \{BA,BBA\}^*$, $w \in T^*$, and $w_1w_2w \Rightarrow^* w$ is derived by $ABBBA \to \varepsilon$.

2.4 Automata

In this section, basic devices for recognizing strings of a given (regular or context-free) language are defined—finite and pushdown automata. These definitions are based on the notation of [Med00a], however, they are equivalent to the so-called delta-notation (see [HU79]).

2.4.1 Finite Automata

Definition 2.4.1. A *finite automaton*, M, is a quintuple

$$M = (Q, \Sigma, \delta, q_0, F),$$

where

- Q is a finite set of *states*,
- Σ is a finite *input* alphabet,
- δ is a finite set of *rules* of the form
$$qw \to p,$$
where $q, p \in Q$ and $w \in \Sigma^*$,
- $q_0 \in Q$ is an *initial* state, and
- F is a set of *final* states.

Definition 2.4.2. Let Ψ be an alphabet of *rule labels* such that $|\Psi| = |\delta|$, and ψ be a bijection from δ to Ψ. For simplicity, to express that ψ maps a rule $qw \to p \in \delta$ to r, where $r \in \Psi$, we write

$$r.qw \to p \in \delta;$$

in other words, $r.qw \to p$ means $\psi(qw \to p) = r$.

A *configuration* of M is any string from $Q\Sigma^*$. For any configuration qwy, where $q \in Q$, $wy \in \Sigma^*$, and any $r.qw \to p \in \delta$, M makes a move from configuration qwy to configuration py according to r, written as

$$qwy \Rightarrow py\,[r].$$

Let χ be any configuration of M. M makes zero moves from χ to χ according to ε, written as

$$\chi \Rightarrow^0 \chi\,[\varepsilon].$$

Let there exist a sequence of configurations $\chi_0, \chi_1, \dots, \chi_n$, for some $n \in \mathbb{N}$, such that $\chi_{i-1} \Rightarrow \chi_i\,[r_i]$, where $r_i \in \Psi$, $i = 1, \dots, n$. Then, M makes n moves from χ_0 to χ_n according to $r_1, \dots, r_n$, symbolically written as

$$\chi_0 \Rightarrow^n \chi_n\,[r_1 \dots r_n].$$

Such a sequence of moves is also called a *computation*. We write $\chi_0 \Rightarrow^+ \chi_n\,[r_1 \dots r_n]$ if $\chi_0 \Rightarrow^n \chi_n\,[r_1 \dots r_n]$, for some $n \in \mathbb{N}$. Analogously, we write $\chi_0 \Rightarrow^* \chi_n\,[\mu]$ if either $\chi_0 = \chi_n$ and $\mu = \varepsilon$, or $\chi_0 \Rightarrow^+ \chi_n\,[\mu]$, where $\mu = r_1 \dots r_n$, for some $r_1, \dots, r_n \in \Psi$. If $w \in \Sigma^*$ and $q_0 w \Rightarrow^* f\,[\mu]$, for some $f \in F$, then w is *accepted* by M and $q_0 w \Rightarrow^* f\,[\mu]$ is an *acceptance* of w in M.

The *language* of M is defined as

$$L(M) = \{w \in \Sigma^* : q_0 w \Rightarrow^* f\,[\mu] \text{ is an acceptance of } w\}.$$

For a proof of the following theorem see [Med00a].

Theorem 2.4.3. *Let L be a language. L is regular if and only if there is a finite automaton, M, such that $L = L(M)$.*

2.4.2 Pushdown Automata

Pushdown automata represent finite automata extended by a potentially unbounded pushdown store.

Definition 2.4.4. A *pushdown automaton*, M, is a septuple

$$M = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F),$$

where

- Q is a finite set of *states*,
- Σ is a finite *input* alphabet,
- Γ is a finite *pushdown* alphabet,
- δ is a finite set of *rules* of the form

$$Zqw \to \gamma p,$$

  where $q, p \in Q$, $Z \in \Gamma$, $w \in \Sigma^*$, and $\gamma \in \Gamma^*$,
- $q_0 \in Q$ is an *initial* state,
- Z_0 is an *initial pushdown symbol*, and
- F is a set of *final* states.

Definition 2.4.5. Again, let ψ denote the bijection from δ to Ψ, and write

$$r.Zqw \to \gamma p$$

instead of $\psi(Zqw \to \gamma p) = r$.

A *configuration* of M is any string from $\Gamma^* Q \Sigma^*$. For any configuration $xAqwy$, where $x \in \Gamma^*$, $A \in \Gamma$, $q \in Q$, $wy \in \Sigma^*$, and any $r.Aqw \to \gamma p \in \delta$, M makes a move from $xAqwy$ to $x\gamma py$ according to r, written as

$$xAqwy \Rightarrow x\gamma py\,[r].$$

As usual, we define $\Rightarrow^n$, for $n \in \mathbb{N}_0$, $\Rightarrow^+$, and $\Rightarrow^*$. If $w \in \Sigma^*$ and $Z_0q_0w \Rightarrow^* f\,[\mu]$, for some $f \in F$, then w is accepted by M and $Z_0q_0w \Rightarrow^* f\,[\mu]$ is an *acceptance* of w in M.

The *language* of M is defined as

$$L(M) = \{w \in \Sigma^* : Z_0q_0w \Rightarrow^* f\,[\mu] \text{ is an acceptance of } w\}.$$

For a proof of the following theorem see [Med00a].

Theorem 2.4.6. *Let L be a language. L is context-free if and only if there is a pushdown automaton, M, such that $L = L(M)$.*

3

Current Concepts and Results

Undoubtedly, over its history, the most studied languages of the Chomsky hierarchy were regular and context-free languages because of their great practical use. However, a short time after its introduction, some practical applications were discovered for which context-free languages were shown not to be sufficient. According to the Chomsky hierarchy, there was no other way than to consider such languages as being context-sensitive. Nevertheless, most of these languages were quite simple and, therefore, new ways how to describe languages of such types were looked for. These efforts eventually led to the idea of increasing the power of existing formal systems by their regulation.

3.1 Regulated Formal Systems

Formal language theory has paid a great attention to regulated and modified grammars, see [DP89, MŠ05] and papers [Bak72, Boo72, DFP99, Fer00, FP94, GG66, Hib74, Med91, Med94, Med97b]. Specifically, the following regulated grammars have been intensively studied.

- matrix grammars;
- programmed grammars;
- random context grammars;
- scattered context grammars;
- conditional grammars.

The main idea behind the regulation of grammars is to take a simple, well-known grammar that is not as powerful as needed, and to find a new way how to simply increase the power of this grammar. For instance, in most cases of regulated grammars, a context-free grammar is taken as the initial or basic simple grammar. However, other types of grammars of the Chomsky hierarchy were studied as well.

On the other side, the notion of regulated automata is rather new. So far, only two papers by Meduna and Kolář concerning the topic of regulated automata have been published (see [MK00, MK02]). In essence, these automata regulate the use of their rules according to which they make moves by control languages.

Informally, consider a pushdown automaton, M, and a control language, L, over M's rule labels. With L, M accepts a string, w, if and only if L contains a control string according to which M makes a sequence of moves so that it reaches a final configuration after reading w. Moreover, with L, M defines the following three types of accepted languages:

$L(M,L,1)$—the language accepted by final state;
$L(M,L,2)$—the language accepted by empty pushdown;
$L(M,L,3)$—the language accepted by final state and empty pushdown.

For any family of languages, $\mathscr{F}$, set $RPD(\mathscr{F},i) = \{L(M,L,i) : M$ is a pushdown automaton and $L \in \mathscr{F}\}$, where $i = 1,2,3$.

The following results are proved in [MK00].

1. $CF = RPD(REG,1) = RPD(REG,2) = RPD(REG,3)$, and
2. $RE = RPD(LIN,1) = RPD(LIN,2) = RPD(LIN,3)$.

Here, *LIN* denotes the family of context-free languages, where any context-free production $A \to \alpha$ contains no more than one nonterminal in α. Such languages are called *linear*.

In [MK02], some restrictions of regulated pushdown automata are studied. Consider two consecutive moves made by a pushdown automaton, M. If during the first move M does not shorten its pushdown and during the second move it does, then M makes a turn during the second move. A pushdown automaton is one-turn if it makes no more than one turn during any computation starting from an initial configuration. Recall that the one-turn pushdown automata are less powerful than the pushdown automata.

It is proved that one-turn regulated pushdown automata characterize the family of recursively enumerable languages and that this equivalence holds even for some restricted versions of one-turn regulated pushdown automata, such as atomic and reduced one-turn pushdown automata.

During a move, an atomic one-turn regulated pushdown automaton changes a state and, in addition, performs exactly one of the following actions:

1. it pushes a symbol onto the pushdown;
2. it pops a symbol from the pushdown;
3. it reads an input symbol.

A reduced one-turn regulated pushdown automaton has a limited number of some components, such as the number of states, pushdown symbols, or transition rules.

The main result proved in [MK02] is that every recursively enumerable language is accepted by an atomic reduced one-turn regulated pushdown automaton in terms of (A) acceptance by final state, (B) acceptance by empty pushdown, and (C) acceptance by final state and empty pushdown. More specifically, it proves that atomic one-turn pushdown automata with no more than one state and two pushdown symbols regulated by linear languages characterize the family of recursively enumerable languages.

One of the main aims of this monograph is to contribute to this topic.

3.2 Descriptional Complexity of Grammars

The second part of this monograph is concerning descriptional complexity of grammars. As mentioned above, this is a vivid trend of formal language theory that has recently so intensified that an annual international conference *Descriptional Complexity of Formal Systems* is held to discuss this specific topic (see [MPPW05, LRCP06, GP07] for its latest proceedings). The following list presents the known results concerning descriptional complexity of partially parallel grammars and grammars regulated by context conditions. It is known that every recursively enumerable language is generated

(1) by a scattered context grammar with no more than five nonterminal symbols and two non-context-free productions (see [Vas05]);
(2) by a multisequential grammar with no more than six nonterminal symbols (see [Med97c]);
(3) by a multicontinuous grammar with no more than six nonterminal symbols (see [Med98a]);
(4) by a context-conditional grammar of degree $(1,1)$ (however, the number of conditional productions and nonterminals is not limited, see [DP89, May72, Sal73]);
(5) by a generalized forbidding grammar of degree two with no more than thirteen conditional productions and fifteen nonterminal symbols (see [MŠ03]); and
(6) by a simple semi-conditional grammar of degree $(2,1)$ with no more than ten conditional productions and twelve nonterminal symbols (see [Vas05]).

Part I

Self-Regulating Automata

4

Self-Regulating Automata

This chapter studies self-regulating finite and pushdown automata. The first section introduces self-regulating finite and pushdown automata and defines two variants how they accept an input string—so-called first-move and all-move self-regulating finite and pushdown automata. The second section studies first-move and all-move self-regulating finite automata and describes their power with respect to the number of turns. Then, some closure properties of families of languages accepted by these automata are studied. Finally, both variants of self-regulating finite automata are compared. The last section of this chapter studies self-regulating pushdown automata. Although the first-move self-regulating pushdown automata are introduced, the question of their power is an open problem.

4.1 Definitions and Examples

This section introduces self-regulating finite and pushdown automata and two ways how they accept an input string.

Consider a finite (pushdown) automaton with a selected state, so-called turn state, and with a finite relation on the alphabet of rule labels. Such an automaton is said to be a self-regulating finite (pushdown) automaton.

Definition 4.1.1. Let $N = (Q, \Sigma, \delta, q_0, F)$ be a finite ($N = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ be a pushdown) automaton. A *self-regulating finite (pushdown) automaton*, SFA (SPDA), M, is a triple

$$M = (N, q_t, R),$$

where

1. $q_t \in Q$ is a *turn state*, and
2. $R \subseteq \Psi \times \Psi$ is a finite relation on the alphabet of N's rule labels, Ψ (see Definition 2.4.2).

Notation 1. Let $N = (Q, \Sigma, \delta, q_0, F)$ be a finite automaton. The self-regulating finite automaton

$$M = (N, q_t, R)$$

is, to clarify the components of N, written as

$$M = (Q, \Sigma, \delta, q_0, q_t, F, R)$$

from now on. Analogously for self-regulating pushdown automata.

4.1.1 Self-Regulating Finite Automata

The main idea of the self-regulating finite automata is as follows. Consider a finite automaton, N. This automaton starts its computation in the start state and then, during its computation, reads the input string and, accordingly, goes from a state to another one. If, having read the whole input string, the computation ends in a final state, the input is accepted; otherwise, the input is rejected. A self-regulating finite automaton, $M = (N, q_t, R)$, is a finite automaton that behaves as follows. M starts in the start state and while it does not reach the turn state, it reads the input, moves from a state to another state according to the applied rule and records the rule. If M reaches the turn state for the first time, i.e. state q_t is, for the first time, the current state of M, the automaton makes a turn. It means that M, in addition, starts to read the recorded sequence of rules, and the computation proceeds according to the relation R. More precisely, M reads an input symbol a, reads the first recorded rule, r_1, of the sequence of rules, $r_1 r_2 \dots r_k$, goes from the current state to another one according to a rule s_1 such that $(r_1, s_1) \in R$, replaces r_1 with s_1, and the next recorded rule is read, r_2. After the whole sequence $r_1 r_2 \dots r_k$ has been read, M makes a turn again or finishes its computation. Note that only in case of the first turn the current state is required to be the turn state q_t. If M makes $n \in \mathbb{N}_0$ turns during its computation, it is called an n-turn self-regulating finite automaton.

Now, let us formally define two variants self-regulating finite automata can accepted an input string. The first variant are so-called n-turn first-move self-regulating finite automata. The phrase "first-move" means that only the first rule applied after a turn is required to be in R with the first rule of the current recorded sequence of rules.

Definition 4.1.2. Let $n \in \mathbb{N}_0$ and

$$M = (Q, \Sigma, \delta, q_0, q_t, F, R)$$

be a self-regulating finite automaton. M is said to be an *n-turn first-move self-regulating finite automaton*, n-first-SFA, if M accepts w in the following way. There is an acceptance of the form $q_0 w \Rightarrow^* f\,[\mu]$ such that

$$\mu = r_1^0 \dots r_k^0 r_1^1 \dots r_k^1 \dots r_1^n \dots r_k^n,$$

where $k \in \mathbb{N}$, r_k^0 is the first rule of the form $qx \to q_t$, for some $q \in Q$, $x \in \Sigma^*$, and

$$(r_1^j, r_1^{j+1}) \in R,$$

for all $0 \le j < n$.

The family of languages accepted by n-first-SFAs is denoted by $FIRST_n$.

Example 4.1.3. Consider a one-turn first-move self-regulating finite automaton,

$$M = (\{s,t,f\},\{a,b\},\delta,s,t,\{f\},\{(1,3)\}),$$

with δ containing rules $1.sa \to s$, $2.sa \to t$, $3.tb \to f$, and $4.fb \to f$ (see Fig. 4.1).

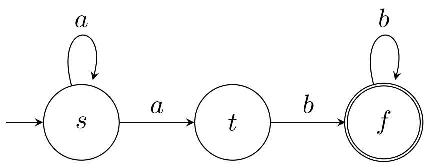

Fig. 4.1. One-turn first-move self-regulating finite automaton M.

With $aabb$, M makes

$$saabb \Rightarrow sabb\,[1] \Rightarrow tbb\,[2] \Rightarrow fb\,[3] \Rightarrow f\,[4].$$

In brief, $saabb \Rightarrow^* f\,[1234]$. Observe that $L(M) = \{a^n b^n : n \geq 1\}$, which belongs to $CF - REG$.

The second variant are so-called n-turn all-move self-regulating finite automata. The phrase "all-move" means that all rules applied after a turn are required to be in R with the corresponding rules of the current recorded sequence of rules.

Definition 4.1.4. Let $n \in \mathbb{N}_0$ and

$$M = (Q,\Sigma,\delta,q_0,q_t,F,R)$$

be a self-regulating finite automaton. M is said to be an *n-turn all-move self-regulating finite automaton*, n-all-SFA, if M accepts w in the following way. There is an acceptance $q_0 w \Rightarrow^* f\,[\mu]$ such that

$$\mu = r_1^0 \ldots r_k^0 r_1^1 \ldots r_k^1 \ldots r_1^n \ldots r_k^n,$$

where $k \in \mathbb{N}$, r_k^0 is the first rule of the form $qx \to q_t$, for some $q \in Q$, $x \in \Sigma^*$, and

$$(r_i^j, r_i^{j+1}) \in R,$$

for all $1 \leq i \leq k$, $0 \leq j < n$.

The family of languages accepted by n-all-SFAs is denoted by ALL_n.

Example 4.1.5. Consider a one-turn all-move self-regulating finite automaton,

$$M = (\{s,t,f\},\{a,b\},\delta,s,t,\{f\},\{(1,4),(2,5),(3,6)\}),$$

with δ containing rules $1.sa \to s$, $2.sb \to s$, $3.s \to t$, $4.ta \to t$, $5.tb \to t$, and $6.t \to f$ (see Fig. 4.2).
With $abab$, M makes

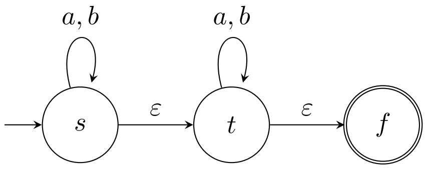

Fig. 4.2. One-turn all-move self-regulating finite automaton M.

$$sabab \Rightarrow sbab\,[1] \Rightarrow sab\,[2] \Rightarrow tab\,[3] \Rightarrow tb\,[4] \Rightarrow t\,[5] \Rightarrow f\,[6]\,.$$

In brief, $sabab \Rightarrow^* f\,[123456]$. Observe that $L(M) = \{ww : w \in \{a,b\}^*\}$, which belongs to $CS - CF$.

4.1.2 Self-Regulating Pushdown Automata

Self-regulating pushdown automata are defined in the same manner as self-regulating finite automata. Formal definitions follow.

Definition 4.1.6. Let $n \in \mathbb{N}_0$ and

$$M = (Q, \Sigma, \Gamma, \delta, q_0, q_t, Z_0, F, R)$$

be a self-regulating pushdown automaton. M is said to be an *n-turn first-move self-regulating pushdown automaton*, n-first-SPDA, if M accepts w in the following way. There is an acceptance $Z_0 q_0 w \Rightarrow^* f\,[\mu]$ such that

$$\mu = r_1^0 \dots r_k^0 r_1^1 \dots r_k^1 \dots r_1^n \dots r_k^n\,,$$

where $k \in \mathbb{N}$, r_k^0 is the first rule of the form $Zqx \to \gamma q_t$, for some $Z \in \Gamma$, $q \in Q$, $x \in \Sigma^*$, $\gamma \in \Gamma^*$, and

$$(r_1^j, r_1^{j+1}) \in R\,,$$

for all $0 \le j < n$.

The family of languages accepted by n-first-SPDAs is denoted by $FIRST$-$SPDA_n$.

Definition 4.1.7. Let $n \in \mathbb{N}_0$ and

$$M = (Q, \Sigma, \Gamma, \delta, q_0, q_t, Z_0, F, R)$$

be a self-regulating pushdown automaton. M is said to be an *n-turn all-move self-regulating pushdown automaton*, n-all-SPDA, if M accepts w in the following way. There is an acceptance $Z_0 q_0 w \Rightarrow^* f\,[\mu]$ such that

$$\mu = r_1^0 \dots r_k^0 r_1^1 \dots r_k^1 \dots r_1^n \dots r_k^n\,,$$

where $k \in \mathbb{N}$, r_k^0 is the first rule of the form $Zqx \to \gamma q_t$, for some $Z \in \Gamma$, $q \in Q$, $x \in \Sigma^*$, $\gamma \in \Gamma^*$, and

$$(r_i^j, r_i^{j+1}) \in R\,,$$

for all $1 \le i \le k$, $0 \le j < n$.

The family of languages accepted by n-all-SPDAs is denoted by ALL-$SPDA_n$.

4.2 Self-Regulating Finite Automata

In this section, the main results concerning self-regulating finite automata are proved.

4.2.1 First-Move Self-Regulating Finite Automata

This section proves the identity between the family of languages accepted by n-turn first-move self-regulating finite automata and the family of languages generated by $(n+1)$-parallel right linear grammars. To do so, a special form of parallel right linear grammars is needed. First, however, parallel right linear grammars are defined (see [RW73, RW75, Woo73, Woo75]).

Definition 4.2.1. For $n \in \mathbb{N}$, an *n-parallel right linear grammar*, n-PRLG, is an $(n+3)$-tuple

$$G = (N_1, \dots, N_n, T, S, P)\,,$$

where

- N_i, $1 \leq i \leq n$, are pairwise disjoint nonterminal alphabets,
- T is a terminal alphabet,
- $S \notin N = N_1 \cup \ldots \cup N_n$ is the start symbol, $N \cap T = \emptyset$, and
- P is a finite set of productions of the following three forms:

  1. $S \to X_1 \ldots X_n$, $\quad X_i \in N_i$, $1 \leq i \leq n$;
  2. $X \to wY$, $\quad X, Y \in N_i$, for some $1 \leq i \leq n$, $w \in T^*$;
  3. $X \to w$, $\quad X \in N$, $w \in T^*$.

For $x, y \in (N \cup T \cup \{S\})^*$, $x \Rightarrow y$ if and only if

1. either $x = S$ and $S \to y \in P$, or
2. $x = y_1 X_1 \ldots y_n X_n$, $y = y_1 x_1 \ldots y_n x_n$, where $y_i \in T^*$, $X_i \in N_i$, and $X_i \to x_i \in P$, for $i = 1, \dots, n$.

Relations $\Rightarrow^n$, for $n \in \mathbb{N}_0$, $\Rightarrow^+$, and $\Rightarrow^*$ are defined as usual.

The language generated by an n-parallel right linear grammar, G, is defined as

$$L(G) = \{w \in T^* : S \Rightarrow^* w\}\,.$$

A language, L, is an *n-parallel right linear language*, n-PRLL, if there is an n-PRLG, G, such that $L = L(G)$. The family of n-PRLLs is denoted by R_n.

Definition 4.2.2. Let $G = (N_1, \dots, N_n, T, S, P)$ be an n-PRLG, for some $n \in \mathbb{N}$, and let $i = 1, \dots, n$. By the *ith component* of G we understand a one-PRLG

$$G = (N_i, T, S', P')\,,$$

where P' contains productions of the following forms:

1. $S' \to X_i$ $\quad$ if $S \to X_1 \ldots X_n \in P$, $X_i \in N_i$;
2. $X \to wY$ $\quad$ if $X \to wY \in P$ and $X, Y \in N_i$;
3. $X \to w$ $\quad$ if $X \to w \in P$ and $X \in N_i$.

The following special form of parallel right linear grammars is needed to prove the main results.

Lemma 4.2.3. *For every n-PRLG $G = (N_1, \ldots, N_n, T, S, P)$, there is an equivalent n-PRLG*

$$G' = (N'_1, \ldots, N'_n, T, S, P')$$

that satisfies:

1. *if $S \to X_1 \ldots X_n \in P'$, then X_i does not occur on the right-hand side of any production, for $i = 1, \ldots, n$;*
2. *if $S \to \alpha$, $S \to \beta \in P'$ and $\alpha \neq \beta$, then $alph(\alpha) \cap alph(\beta) = \emptyset$.*

Proof. If G does not satisfy conditions from the lemma, then we will construct a new n-PRLG $G' = (N'_1, \ldots, N'_n, T, S, P')$, where P' contains all productions of the form $X \to \beta \in P$, $X \neq S$, and $N_j \subseteq N'_j$, for $j = 1, \ldots, n$. For each production $S \to X_1 \ldots X_n \in P$, we add new nonterminals $Y_j \notin N'_j$ into N'_j, and productions include $S \to Y_1 \ldots Y_n$ and $Y_j \to X_j$ in P', for $j = 1, \ldots, n$. Clearly,

$$S \Rightarrow_G X_1 \ldots X_n \text{if and only if } S \Rightarrow_{G'} Y_1 \ldots Y_n \Rightarrow X_1 \ldots X_n\,.$$

Thus, $L(G) = L(G')$. □

The following lemma says that every language generated by an n-parallel right linear grammar can be accepted by an $(n-1)$-turn first-move self-regulating finite automaton. Thus, first-move self-regulating finite automata are at least as powerful as parallel right linear grammars.

Lemma 4.2.4. *Let G be an n-PRLG. Then, there is an $(n-1)$-first-SFA, M, such that $L(G) = L(M)$.*

The basic idea of the proof is that M is divided into n parts (see Fig. 4.3). The ith part represents a finite automaton accepting the language of G's ith component, and R also connects the ith part to the $(i+1)$st part as depicted in Fig. 4.3.

Proof. Without loss of generality, we can assume that $G = (N_1, \ldots, N_n, T, S, P)$ is in the form from Lemma 4.2.3. Construct an $(n-1)$-first-SFA

$$M = (Q, T, \delta, q_0, q_t, F, R)\,,$$

where

$$\begin{aligned}
Q &= \{q_0, \ldots, q_n\} \cup N,\ N = N_1 \cup \ldots \cup N_n, \text{ and } \{q_0, q_1, \ldots, q_n\} \cap N = \emptyset,\\
F &= \{q_n\},\\
\delta &= \begin{cases} \{q_i \to X_{i+1} : S \to X_1 \ldots X_n \in P, 0 \leq i < n\} \cup \\ \{Xw \to Y : X \to wY \in P\} \cup \\ \{Xw \to q_i : X \to w \in P, w \in T^*, X \in N_i, 1 \leq i \leq n\}, \end{cases}\\
q_t &= q_1,\\
\Psi &= \delta \text{ with the identity map, and}\\
R &= \{(q_i \to X_{i+1}, q_{i+1} \to X_{i+2}) : S \to X_1 \ldots X_n \in P, 0 \leq i \leq n-2\}.
\end{aligned}$$

We prove that $L(G) = L(M)$. To prove that $L(G) \subseteq L(M)$, consider a derivation of w in G and construct an acceptance of w in M depicted in Fig. 4.3. This figure clearly

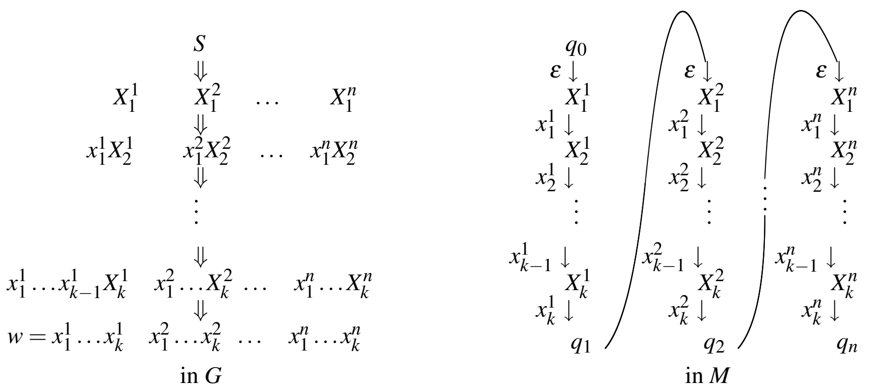

Fig. 4.3. A derivation of w in G and the corresponding acceptance of w in M.

demonstrates the fundamental idea behind this part of the proof; its complete and rigorous version is lengthy and left to the reader. Thus, for each derivation $S \Rightarrow^* w$, $w \in T^*$, there is an acceptance of w in M.

To prove that $L(M) \subseteq L(G)$, let $w \in L(M)$, and consider an acceptance of w in M. Observe that the acceptance is of the form depicted on the right-hand side of Fig. 4.3. It means that the number of steps M made from q_{i-1} to q_i is the same as from q_i to q_{i+1} since the only rule in the relation with $q_{i-1} \to X_1^i$ is the rule $q_i \to X_1^{i+1}$. Moreover, M can never come back to a state corresponding to a previous component. (By a component of M, we mean the finite automaton $M_i = (Q, \Sigma, \delta, q_{i-1}, \{q_i\})$, for $1 \leq i \leq n$.) Now, construct a derivation of w in G. By Lemma 4.2.3, we have $|\{X : (q_i \to X_1^{i+1}, q_{i+1} \to X) \in R\}| = 1$, for all $0 \leq i < n-1$. Thus, $S \to X_1^1 X_1^2 \dots X_1^n \in P$. Moreover, if $X_j^i x_j^i \to X_{j+1}^i$, we apply $X_j^i \to x_j^i X_{j+1}^i \in P$, and if $X_k^i x_k^i \to q_i$, we apply $X_k^i \to x_k^i \in P$, $1 \leq i \leq n$, $1 \leq j < k$.

Hence, Lemma 4.2.4 holds. □

The following lemma says that every language accepted by an n-turn first-move self-regulating finite automaton can be generated by an $(n+1)$-parallel right linear grammar. Thus, parallel right linear grammars are at least as powerful as first-move self-regulating finite automata.

Lemma 4.2.5. *Let M be an n-first-SFA. There is an $(n+1)$-PRLG, G, such that $L(G) = L(M)$.*

Proof. Let $M = (Q, \Sigma, \delta, q_0, q_t, F, R)$. Consider

$$G = (N_0, \dots, N_n, \Sigma, S, P),$$

where

$$\begin{aligned}
N_i &= \big(Q(\Sigma \cup \{\varepsilon\})^l \times Q \times \{i\} \times Q\big) \cup (Q \times \{i\} \times Q),\\
l &= \max\{|w| : qw \to p \in \delta\},\ 0 \le i \le n, \text{ and}\\
P &= \{S \to [q_0x_0, q^0, 0, q_t][q_tx_1, q^1, 1, q_{i_1}][q_{i_1}x_2, q^2, 2, q_{i_2}] \dots [q_{i_{n-1}}x_n, q^n, n, q_{i_n}] :\\
&\qquad r_0.q_0x_0 \to q^0,\ r_1.q_tx_1 \to q^1,\ r_2.q_{i_1}x_2 \to q^2, \dots,\ r_n.q_{i_{n-1}}x_n \to q^n \in \delta,\\
&\qquad (r_0, r_1), (r_1, r_2), \dots, (r_{n-1}, r_n) \in R,\ q_{i_n} \in F\} \cup\\
&\quad \{[px, q, i, r] \to x[q, i, r]\} \cup\\
&\quad \{[q, i, q] \to \varepsilon : q \in Q\} \cup\\
&\quad \{[q, i, p] \to w[q', i, p] : qw \to q' \in \delta\}.
\end{aligned}$$

We prove that $L(G) = L(M)$. To prove that $L(G) \subseteq L(M)$, observe that we make $n+1$ copies of M and go through them similarly to Fig. 4.3. Consider a derivation of w in G. Then, in greater detail, this derivation is of the form

$$\begin{aligned}
S &\Rightarrow [q_0x_0^0, q_1^0, 0, q_t][q_tx_0^1, q_1^1, 1, q_{i_1}] \dots [q_{i_{n-1}}x_0^n, q_1^n, n, q_{i_n}]\\
&\Rightarrow x_0^0[q_1^0, 0, q_t]x_0^1[q_1^1, 1, q_{i_1}] \dots x_0^n[q_1^n, n, q_{i_n}]\\
&\Rightarrow x_0^0x_1^0[q_2^0, 0, q_t]x_0^1x_1^1[q_2^1, 1, q_{i_1}] \dots x_0^nx_1^n[q_2^n, n, q_{i_n}]\\
&\ \vdots\\
&\Rightarrow x_0^0x_1^0 \dots x_k^0[q_t, 0, q_t]x_0^1x_1^1 \dots x_k^1[q_{i_1}, 1, q_{i_1}] \dots x_0^nx_1^n \dots x_k^n[q_{i_n}, n, q_{i_n}]\\
&\Rightarrow x_0^0x_1^0 \dots x_k^0x_0^1x_1^1 \dots x_k^1 \dots x_0^nx_1^n \dots x_k^n
\end{aligned} \tag{4.1}$$

and

$$\begin{gathered}
r_0.q_0x_0^0 \to q_1^0,\ r_1.q_tx_0^1 \to q_1^1,\ r_2.q_{i_1}x_0^2 \to q_1^2, \dots,\ r_n.q_{i_{n-1}}x_0^n \to q_1^n \in \delta,\\
(r_0, r_1),\ (r_1, r_2), \dots,\ (r_{n-1}, r_n) \in R,
\end{gathered}$$

and $q_{i_n} \in F$.

Thus, the list of rules used in the acceptance of w in M is

$$\begin{aligned}
\mu = {} & (q_0x_0^0 \to q_1^0)(q_1^0x_1^0 \to q_2^0) \dots (q_k^0x_k^0 \to q_t)\\
& (q_tx_0^1 \to q_1^1)(q_1^1x_1^1 \to q_2^1) \dots (q_k^1x_k^1 \to q_{i_1})\\
& (q_{i_1}x_0^2 \to q_1^2)(q_1^2x_1^2 \to q_2^2) \dots (q_k^2x_k^2 \to q_{i_2})\\
& \quad \vdots\\
& (q_{i_{n-1}}x_0^n \to q_1^n)(q_1^nx_1^n \to q_2^n) \dots (q_k^nx_k^n \to q_{i_n}).
\end{aligned} \tag{4.2}$$

Now, we prove that $L(M) \subseteq L(G)$. Informally, the acceptance is divided into $n+1$ parts of the same length. Grammar G generates the ith part by the ith component and records the state from which the next component starts.

Let μ be a list of rules used in an acceptance of

$$w = x_0^0x_1^0 \dots x_k^0x_0^1x_1^1 \dots x_k^1 \dots x_0^nx_1^n \dots x_k^n$$

in M of the form (4.2). Then, the derivation of the form (4.1) is the corresponding derivation of w in G since

$$[q_j^i, i, p] \to x_j^i [q_{j+1}^i, i, p] \in P$$

and

$$[q, i, q] \to \varepsilon\,,$$

for all $0 \leq i \leq n$, $1 \leq j < k$.

Hence, Lemma 4.2.5 holds. □

The first main result of this chapter is that first-move self-regulating finite automata are as powerful as parallel right linear grammars.

Theorem 4.2.6. *For all $n \in \mathbb{N}_0$, $FIRST_n = R_{n+1}$.*

Proof. This proof follows from Lemmas 4.2.4 and 4.2.5. □

Corollary 4.2.7. *The following statements hold true.*

1. *$REG = FIRST_0 \subset FIRST_1 \subset FIRST_2 \subset \ldots \subset CS$.*
2. *$FIRST_1 \subset CF$.*
3. *$FIRST_2 \not\subseteq CF$.*
4. *$CF \not\subseteq FIRST_n$ for any $n \in \mathbb{N}_0$.*
5. *For all $n \in \mathbb{N}_0$, $FIRST_n$ is closed under union, finite substitution, homomorphism, intersection with a regular language, and right quotient with a regular language.*
6. *For all $n \in \mathbb{N}$, $FIRST_n$ is not closed under intersection and complement.*

Proof. Recall the following statements proved in [RW75]:

- $REG = R_1 \subset R_2 \subset R_3 \subset \ldots \subset CS$.
- $R_2 \subset CF$.
- $CF \not\subseteq R_n$, $n \in \mathbb{N}$.
- For all $n \in \mathbb{N}$, R_n is closed under union, finite substitution, homomorphism, intersection with a regular language, and right quotient with a regular language.
- For all $n \in \mathbb{N} - \{1\}$, R_n is not closed under intersection and complement.

These statements and Theorem 4.2.6 imply statements 1, 2, 4, 5, 6 of Corollary 4.2.7. Moreover, observe that $\{a^n b^n c^{2n} : n \in \mathbb{N}_0\} \in FIRST_2 - CF$, which proves 3. □

Theorem 4.2.8. *For all $n \in \mathbb{N}$, $FIRST_n$ is not closed under inverse homomorphism.*

Proof. For $n = 1$, let $L = \{a^k b^k : k \in \mathbb{N}\}$, and let the homomorphism $h : \{a, b, c\}^* \to \{a, b\}^*$ be defined as $h(a) = a$, $h(b) = b$, and $h(c) = \varepsilon$. Then, it is not hard to see that $L \in FIRST_1$. However, we prove that

$$L' = h^{-1}(L) \cap c^* a^* b^* = \{c^* a^k b^k : k \in \mathbb{N}\} \notin FIRST_1\,.$$

Assume that L' is in $FIRST_1$. Then, by Theorem 4.2.6, there is a two-PRLG

$$G = (N_1, N_2, T, S, P)$$

such that $L(G) = L'$. Let $k > |P| \cdot \max\{|w| : X \to wY \in P\}$. Consider a derivation of $c^k a^k b^k \in L'$. The second component can generate only finitely many as; otherwise, it derives $\{a^k b^n : k < n\}$, which is not regular. Analogously, the first component generates only finitely many bs. Therefore, the first component generates any number of as, and the second component generates any number of bs. Moreover, there is a derivation of the form $X \Rightarrow^m X$, for some $X \in N_2$, and $m \in \mathbb{N}$, used in the derivation in the second component. In the first component, there is a derivation $A \Rightarrow^l a^s A$, for some $A \in N_1$, and $s, l \in \mathbb{N}$. Then, we can modify the derivation of $c^k a^k b^k$ so that in the first component, we repeat the cycle $A \Rightarrow^l a^s A$ $(m+1)$-times, and in the second component, we repeat the cycle $X \Rightarrow^m X$ $(l+1)$-times. The derivations of both components have the same length—the added cycles are of length ml, and the rest is of the same length as in the derivation of $c^k a^k b^k$. Therefore, we have derived $c^k a^r b^k$, where $r > k$, which is not in L'—a contradiction.

For $n > 1$, the proof is analogous and left to the reader. □

Corollary 4.2.9. *For all $n \in \mathbb{N}$, $FIRST_n$ is not closed under concatenation. Therefore, it is not closed under Kleene closure either.*

Proof. For $n = 1$, let $L_1 = \{c\}^*$ and $L_2 = \{a^k b^k : k \in \mathbb{N}\}$. Then, $L_1 L_2 = \{c^* a^k b^k : k \in \mathbb{N}\}$. Analogously for $n > 1$. Moreover, let $L = L_1 \cup L_2$. Then, $L^* \cap \{c\}^*\{a\}^+\{b\}^+ = L_1 L_2$. □

4.2.2 All-Move Self-Regulating Finite Automata

This section discusses n-turn all-move self-regulating finite automata. It proves that the family of languages accepted by n-turn all-move self-regulating finite automata coincides with the family of languages generated by n-right linear simple matrix grammars.

Definition 4.2.10. For $n \in \mathbb{N}$, an *n-right linear simple matrix grammar*, n-RLSMG, is an $(n+3)$-tuple

$$G = (N_1, \ldots, N_n, T, S, P),$$

where

- N_i, $1 \leq i \leq n$, are pairwise disjoint nonterminal alphabets,
- T is a terminal alphabet,
- $S \notin N = N_1 \cup \ldots \cup N_n$ is the start symbol, $N \cap T = \emptyset$, and
- P is a finite set of matrix rules. Any matrix rule can be in one of the following three forms:

  1. $[S \to X_1 \ldots X_n]$, $\quad X_i \in N_i$, $1 \leq i \leq n$;
  2. $[X_1 \to w_1 Y_1, \ldots, X_n \to w_n Y_n]$, $\quad w_i \in T^*$, $X_i, Y_i \in N_i$, $1 \leq i \leq n$;
  3. $[X_1 \to w_1, \ldots, X_n \to w_n]$, $\quad X_i \in N_i$, $w_i \in T^*$, $1 \leq i \leq n$.

Let m be a matrix, then $m[i]$ denotes the ith rule of m.

For $x, y \in (N \cup T \cup \{S\})^*$, $x \Rightarrow y$ if and only if

1. either $x = S$ and $[S \to y] \in P$,

2. or $x = y_1 X_1 \dots y_n X_n$, $y = y_1 x_1 \dots y_n x_n$, where $y_i \in T^*$, $X_i \in N_i$, and $[X_1 \to x_1, \dots, X_n \to x_n] \in P$, for $i = 1, \dots, n$.

We define $\Rightarrow^n$, for $n \in \mathbb{N}_0$, $x \Rightarrow^+ y$, and $x \Rightarrow^* y$ as usual.

The language generated by an n-right linear simple matrix grammar, G, is defined as

$$L(G) = \{w \in T^* : S \Rightarrow^* w\} .$$

A language, L, is an *n-right linear simple matrix language*, n-RLSML, if there is an n-RLSMG, G, such that $L = L(G)$. The family of n-RLSMLs is denoted by $R_{[n]}$.

The ith component of an n-RLSMG is defined analogously as in case of parallel right linear grammars.

To prove the main result, the following lemma is needed.

Lemma 4.2.11. *For every n-RLSMG, $G = (N_1, \dots, N_n, T, S, P)$, there is an equivalent n-RLSMG, G', that satisfies:*

1. *if $[S \to X_1 \dots X_n]$, then X_i does not occur on the right-hand side of any rule, for $i = 1, \dots, n$;*
2. *if $[S \to \alpha]$, $[S \to \beta] \in P$ and $\alpha \neq \beta$, then $alph(\alpha) \cap alph(\beta) = \emptyset$;*
3. *for any two matrices $m_1, m_2 \in P$, if $m_1[i] = m_2[i]$, for some $1 \leq i \leq n$, then $m_1 = m_2$.*

Proof. The first two conditions can be proved analogously to Lemma 4.2.3. Suppose that there are matrices m and m' such that $m[i] = m'[i]$, for some $1 \leq i \leq n$. Let $m = [X_1 \to x_1, \dots, X_n \to x_n]$, $m' = [Y_1 \to y_1, \dots, Y_n \to y_n]$. Replace these matrices with matrices $m_1 = [X_1 \to X_1', \dots, X_n \to X_n']$, $m_2 = [X_1' \to x_1, \dots, X_n' \to x_n]$, and $m_1' = [Y_1 \to Y_1'', \dots, Y_n \to Y_n'']$, $m_2' = [Y_1'' \to y_1, \dots, Y_n'' \to y_n]$, where X_i', Y_i'' are new nonterminals, for all $i = 1, \dots, n$. These new matrices satisfy condition 3. Repeat this replacement until the resulting grammar satisfies the properties of G' given in this lemma. □

The following lemma says that every language generated by an n-right linear simple matrix grammar can be accepted by an $(n-1)$-turn all-move self-regulating finite automaton. Thus, all-move self-regulating finite automata are at least as powerful as right linear simple matrix grammars.

Lemma 4.2.12. *Let G be an n-RLSMG. There is an $(n-1)$-all-SFA, M, such that $L(G) = L(M)$.*

Proof. Without loss of generality, we can assume that $G = (N_1, \dots, N_n, T, S, P)$ is in the form described in Lemma 4.2.11. Construct an $(n-1)$-all-SFA

$$M = (Q, T, \delta, q_0, q_t, F, R) ,$$

where

$$
\begin{aligned}
&Q = \{q_0, \ldots, q_n\} \cup N,\ N = N_1 \cup \ldots \cup N_n, \text{ and } \{q_0, q_1, \ldots, q_n\} \cap N = \emptyset,\\
&F = \{q_n\},\\
&\delta = \begin{cases} \{q_i \to X_{i+1} : [S \to X_1 \ldots X_n] \in P, 0 \le i < n\} \cup \\ \{X_i w_i \to Y_i : [X_1 \to w_1 Y_1, \ldots, X_n \to w_n Y_n] \in P, 1 \le i \le n\} \cup \\ \{X_i w_i \to q_i : [X_1 \to w_1, \ldots, X_n \to w_n] \in P, w_i \in T^*, 1 \le i \le n\}, \end{cases}\\
&q_t = q_1,\\
&\Psi = \delta \text{ with the identity map, and}\\
&R = \begin{cases} \{(q_i \to X_{i+1}, q_{i+1} \to X_{i+2}) : \\ \qquad [S \to X_1 \ldots X_n] \in P, 0 \le i \le n-2\} \cup \\ \{(X_i w_i \to Y_i, X_{i+1} w_{i+1} \to Y_{i+1}) : \\ \qquad [X_1 \to w_1 Y_1, \ldots, X_n \to w_n Y_n] \in P, 1 \le i < n\} \cup \\ \{(X_i w_i \to q_i, X_{i+1} w_{i+1} \to q_{i+1}) : \\ \qquad [X_1 \to w_1, \ldots, X_n \to w_n] \in P, w_i \in T^*, 1 \le i < n\}. \end{cases}
\end{aligned}
$$

We prove that $L(G) = L(M)$. The proof of the inclusion $L(G) \subseteq L(M)$ is very similar to the proof of the same inclusion of Lemma 4.2.4, so it is left to the reader.

To prove that $L(M) \subseteq L(G)$, consider $w \in L(M)$ and an acceptance of w in M. As in Lemma 4.2.4, the derivation looks like the one depicted on the right-hand side of Fig. 4.3. We generate w in G as follows. By Lemma 4.2.11, there is matrix $[S \to X_1^1 X_1^2 \ldots X_1^n]$ in P. Moreover, if $X_j^i x_j^i \to X_{j+1}^i$, $1 \le i \le n$, then

$$(X_j^i \to x_j^i X_{j+1}^i, X_j^{i+1} \to x_j^{i+1} X_{j+1}^{i+1}) \in R,$$

for $1 \le i < n$, $1 \le j < k$. We apply

$$[X_j^1 \to x_j^1 X_{j+1}^1, \ldots, X_j^n \to x_j^n X_{j+1}^n]$$

from P. If $X_k^i x_k^i \to q_i$, $1 \le i \le n$, then

$$(X_k^i \to x_k^i, X_k^{i+1} \to x_k^{i+1}) \in R,$$

for $1 \le i < n$, and we apply

$$[X_k^1 \to x_k^1, \ldots, X_k^n \to x_k^n] \in P.$$

Thus, $w \in L(G)$.

Hence, Lemma 4.2.12 holds. □

The following lemma says that every language accepted by an n-turn all-move self-regulating finite automaton can be generated by an $(n+1)$-right linear simple matrix grammar. Thus, right linear simple matrix grammars are at least as powerful as all-move self-regulating finite automata.

Lemma 4.2.13. *Let M be an n-all-SFA. There is an $(n+1)$-RLSMG, G, such that $L(G) = L(M)$.*

Proof. Let $M = (Q, \Sigma, \delta, q_0, q_t, F, R)$. Consider

$$G = (N_0, \dots, N_n, \Sigma, S, P),$$

where

$$\begin{aligned}
N_i &= \big(Q(\Sigma \cup \{\varepsilon\})^l \times Q \times \{i\} \times Q\big) \cup (Q \times \{i\} \times Q),\\
l &= \max\{|w| : qw \to p \in \delta\},\ 0 \le i \le n, \text{ and}\\
P &= \{[S \to [q_0x_0, q^0, 0, q_t][q_tx_1, q^1, 1, q_{i_1}] \dots [q_{i_{n-1}}x_n, q^n, n, q_{i_n}]] :\\
&\qquad r_0.q_0x_0 \to q^0,\ r_1.q_tx_1 \to q^1, \dots,\ r_n.q_{i_{n-1}}x_n \to q^n \in \delta,\\
&\qquad (r_0, r_1), \dots,\ (r_{n-1}, r_n) \in R,\ q_{i_n} \in F\} \cup\\
&\quad \{[[p_0x_0, q_0, 0, r_0] \to x_0[q_0, 0, r_0], \dots, [p_nx_n, q_n, n, r_n] \to x_n[q_n, n, r_n]]\} \cup\\
&\quad \{[[q_0, 0, q_0] \to \varepsilon, \dots, [q_n, n, q_n] \to \varepsilon] : q_i \in Q, 0 \le i \le n\} \cup\\
&\quad \{[[q_0, 0, p_0] \to w_0[q_0', 0, p_0], \dots, [q_n, n, p_n] \to w_n[q_n', n, p_n]] :\\
&\qquad r_j.q_jw_j \to q_j' \in \delta, 0 \le j \le n, (r_i, r_{i+1}) \in R, 0 \le i < n\}.
\end{aligned}$$

We prove that $L(G) = L(M)$. To prove that $L(G) \subseteq L(M)$, consider a derivation of w in G. Then, the derivation is of the form (4.1) and there are rules

$$r_0.q_0x_0^0 \to q_1^0,\ r_1.q_tx_0^1 \to q_1^1, \dots,\ r_n.q_{i_{n-1}}x_0^n \to q_1^n$$

in δ such that $(r_0, r_1), \dots,\ (r_{n-1}, r_n) \in R$. Moreover, $(r_j^l, r_j^{l+1}) \in R$, where $r_j^l.q_j^lx_j^l \to q_{j+1}^l \in \delta$, and $(r_k^l, r_k^{l+1}) \in R$, where $r_k^l.q_k^lx_k^l \to q_{i_l} \in \delta$, $0 \le l < n$, $1 \le j < k$, q_{i_0} denotes q_t, and $q_{i_n} \in F$. Thus, M accepts w with the list of rules μ of the form (4.2).

To prove that $L(M) \subseteq L(G)$, let μ be a list of rules used in an acceptance of

$$w = x_0^0x_1^0 \dots x_k^0x_0^1x_1^1 \dots x_k^1 \dots x_0^nx_1^n \dots x_k^n$$

in M of the form (4.2). Then, the derivation is of the form (4.1) because

$$[[q_j^0, 0, q_t] \to x_j^0[q_{j+1}^0, 0, q_t], \dots, [q_j^n, n, q_{i_n}] \to x_j^n[q_{j+1}^n, n, q_{i_n}]] \in P,$$

for all $q_j^i \in Q$, $1 \le i \le n$, $1 \le j < k$, and $[[q_t, 0, q_t] \to \varepsilon, \dots, [q_{i_n}, n, q_{i_n}] \to \varepsilon] \in P$.

Hence, Lemma 4.2.13 holds. □

The second main result of this chapter is that all-move self-regulating finite automata are as powerful as right linear simple matrix grammars.

Theorem 4.2.14. *For all $n \in \mathbb{N}_0$, $ALL_n = R_{[n+1]}$.*

Proof. This proof follows from Lemmas 4.2.12 and 4.2.13. □

Corollary 4.2.15. *The following statements hold:*

1. $REG = ALL_0 \subset ALL_1 \subset ALL_2 \subset \dots \subset CS$.
2. $ALL_1 \not\subseteq CF$.
3. $CF \not\subseteq ALL_n$, *for every* $n \in \mathbb{N}_0$.

4. *For all $n \in \mathbb{N}_0$, ALL_n is closed under union, concatenation, finite substitution, homomorphism, intersection with a regular language, and right quotient with a regular language.*
5. *For all $n \in \mathbb{N}$, ALL_n is not closed under intersection, complement, and Kleene closure.*

Proof. Recall the following statements proved in [Woo75]:

- $REG = R_{[1]} \subset R_{[2]} \subset R_{[3]} \subset \ldots \subset CS$.
- For all $n \in \mathbb{N}$, $R_{[n]}$ is closed under union, finite substitution, homomorphism, intersection with a regular language, and right quotient with a regular language.
- For all $n \in \mathbb{N} - \{1\}$, $R_{[n]}$ is not closed under intersection and complement.

Furthermore, recall statements proved in [Sir69] and [Sir71]:

- For all $n \in \mathbb{N}$, $R_{[n]}$ is closed under concatenation.
- For all $n \in \mathbb{N} - \{1\}$, $R_{[n]}$ is not closed under Kleene closure.

These statements and Theorem 4.2.14 imply statements 1, 4, and 5 of Corollary 4.2.15. Moreover, observe that $\{ww : w \in \{a,b\}^*\} \in ALL_1 - CF$ (see *Example 4.1.5*), which proves 2. Finally, let $L = \{wcw^R : w \in \{a,b\}^*\}$. In [DP89, Theorem 1.5.2], there is a proof that $L \notin R_{[n]}$, for any $n \in \mathbb{N}$. Thus, 3 follows from Theorem 4.2.14. □

Theorem 4.2.16, given next, follows from Theorem 4.2.14 and from Corollary 3.3.3 in [Sir71]. However, Corollary 3.3.3 in [Sir71] is not proved effectively. We next prove Theorem 4.2.16 effectively.

Theorem 4.2.16. *ALL_n is closed under inverse homomorphism, for all $n \in \mathbb{N}_0$.*

The basic idea of the proof is to simulate the derivation of a one-all-SFA, M, as follows. If M reads a, the simulation proceeds as if M reads $h(a)$, where h is a given homomorphism. Since $h(a)$ is a string, we store $h(a)$ in the state of the simulating automaton and then, in the state, simulate the reading of $h(a)$. However, the automaton can make a turn while a piece of $h(a)$ is still stored in the state. This string, in the proof denoted by y, must be carried over.

Proof. For $n = 1$, let $M = (Q, \Sigma, \delta, q_0, q_t, F, R)$ be a one-all-SFA, and let $h : \Delta^* \to \Sigma^*$ be a homomorphism. Construct a one-all-SFA

$$M' = (Q', \Delta, \delta', q_0', q_t', \{q_f'\}, R')$$

accepting $h^{-1}(L(M))$ as follows. Denote $k = \max\{|w| : qw \to p \in \delta\} + \max\{|h(a)| : a \in \Delta\}$. Let

$$Q' = q_0' \cup \{[x, q, y] : x, y \in \Sigma^*, |x|, |y| \le k, q \in Q\}.$$

Initially, set δ' and R' to $\emptyset$. Then, extend δ' and R' by performing 1 through 5, where δ' contains exactly the rules used in R'.

1. For $y \in \Sigma^*$, $|y| \le k$, add
   $(q_0' \to [\varepsilon, q_0, y], q_t' \to [y, q_t, \varepsilon])$ to R';

2. For $A \in Q'$, $q \neq q_t$, add
   $([x,q,y]a \to [xh(a),q,y], A \to A)$ to R';
3. For $A \in Q'$, add
   $(A \to A, [x,q,\varepsilon]a \to [xh(a),q,\varepsilon])$ to R';
4. For $(qx \to p, q'x' \to p') \in R$, $q \neq q_t$, add
   $([xw,q,y] \to [w,p,y], [x'w',q',\varepsilon] \to [w',p',\varepsilon])$ to R';
5. For $q_f \in F$, add
   $([y,q_t,y] \to q'_t, [\varepsilon,q_f,\varepsilon] \to q'_f)$ to R'.

In essence, M' simulates M in the following way. In a state of the form $[x,q,y]$, the three components have the following meaning:

- $x = h(a_1 \dots a_n)$, where $a_1 \dots a_n$ is the input string that M' has already read;
- q is the current state of M;
- y is the suffix remaining as the first component of the state that M' enters during a turn; y is thus obtained when M' reads the last symbol right before the turn occurs in M; M reads y after the turn.

More precisely, $h(w) = w_1 y w_2$, where w is an input string, w_1 is accepted by M before making the turn, i.e. from q_0 to q_t, and yw_2 is accepted by M after making the turn, i.e. from q_t to $q_f \in F$.

For $n > 1$, the proof is analogous. □

4.2.3 Language Families Accepted by n-first-SFAs and n-all-SFAs

This section compares the family of languages accepted by n-turn first-move self-regulating finite automata with the family of languages accepted by n-turn all-move self-regulating finite automata.

Theorem 4.2.17. *For all $n \in \mathbb{N}$, $FIRST_n \subset ALL_n$.*

Proof. In [RW75] and [Woo75], it is proved that for all $n \in \mathbb{N} - \{1\}$, $R_n \subset R_{[n]}$. The proof of Theorem 4.2.17 thus follows from Theorems 4.2.6 and 4.2.14. □

Theorem 4.2.18. *$FIRST_n \not\subseteq ALL_{n-1}$, for all $n \in \mathbb{N}$.*

Proof. It is easy to see that $L = \{a_1^k a_2^k \dots a_{n+1}^k : k \in \mathbb{N}\} \in FIRST_n = R_{n+1}$. However, $L \notin ALL_{n-1} = R_{[n]}$ (see Lemma 1.5.6 in [DP89]). □

Lemma 4.2.19. *For each regular language, L, language $\{w^n : w \in L\} \in ALL_{n-1}$.*

Proof. Let $L = L(M)$, where M is a finite automaton. Make n copies of M. Rename their states so all the sets of states are pairwise disjoint. In this way, also rename the states in the rules of each of these n automata; however, keep the labels of the rules unchanged. For each rule label r, include (r,r) into R. As a result, we obtain an n-turn all-move self-regulating finite automaton that accepts $\{w^n : w \in L\}$. □

Theorem 4.2.20. *$ALL_n - FIRST \neq \emptyset$, $FIRST = \bigcup_{m=1}^{\infty} FIRST_m$, for all $n \in \mathbb{N}$.*

Proof. By induction on $n \in \mathbb{N}$, we prove that language

$$L = \{(cw)^{n+1} : w \in \{a,b\}^*\} \notin FIRST\,.$$

From Lemma 4.2.19, $L \in ALL_n$.

Basis: For $n = 1$, let G be an m-PRLG generating L, for some positive integer m. Consider a sufficiently large string $cw_1cw_2 \in L$ such that $w_1 = w_2 = a^{n_1}b^{n_2}$, $n_2 > n_1 > 1$. Then, there is a derivation of the form

$$\begin{aligned} S &\Rightarrow^p \\ x_1A_1x_2A_2\ldots x_mA_m &\Rightarrow^k x_1y_1A_1x_2y_2A_2\ldots x_my_mA_m \end{aligned} \tag{4.3}$$

in G, where cycle (4.3) generates more than one a in w_1. The derivation continues as

$$\begin{aligned} &x_1y_1A_1x_2y_2A_2\ldots x_my_mA_m \Rightarrow^r \\ &\quad x_1y_1z_1B_1\ldots x_my_mz_mB_m \Rightarrow^l x_1y_1z_1u_1B_1\ldots x_my_mz_mu_mB_m \\ &\text{(cycle (4.4) generates no } a\text{s)} \Rightarrow^s cw_1cw_2\,. \end{aligned} \tag{4.4}$$

Next, modify the left derivation, the derivation in components generating cw_1, so that the a-generating cycle (4.3) is repeated $(l+1)$-times. Similarly, modify the right derivation, the derivation in the other components, so that the no-a-generating cycle (4.4) is repeated $(k+1)$-times. Thus, the modified left derivation is of length

$$p+k(l+1)+r+l+s = p+k+r+l(k+1)+s\,,$$

which is the length of the modified right derivation. Moreover, the modified left derivation generates more as in w_1 than the right derivation in w_2—a contradiction.

Induction step: Suppose that the theorem holds for $n \geq 2$, and consider $n+1$. Let

$$\{(cw)^{n+1} : w \in \{a,b\}^*\} \in FIRST_l\,,$$

for some $l \in \mathbb{N}$. As $FIRST_l$ is closed under right quotient with a regular language, and $\{cw : w \in \{a,b\}^*\}$ is regular, we obtain $\{(cw)^n : w \in \{a,b\}^*\} \in FIRST_l \subseteq FIRST$—a contradiction. □

Fig. 4.4 summarizes the language families discussed so far.

4.3 Self-Regulating Pushdown Automata

The previous section has discussed self-regulating finite automata. Next section discusses self-regulating pushdown automata.

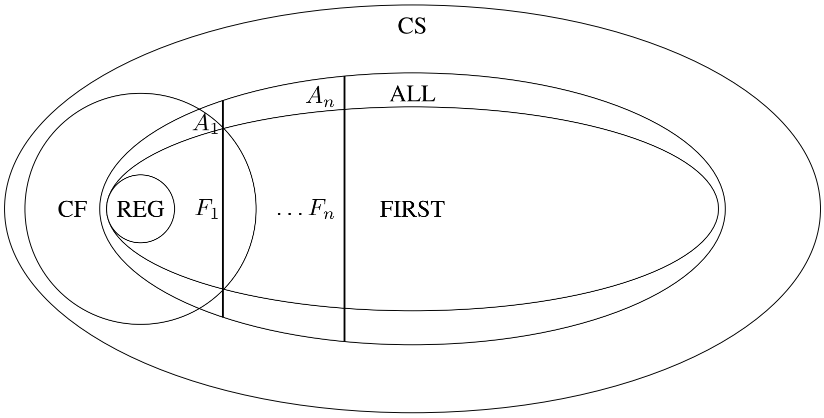

Fig. 4.4. The hierarchy of languages. Here, F_n stands for $FIRST_n$, and A_n for ALL_n.

4.3.1 All-Move Self-Regulating Pushdown Automata

It is easy to see that an all-move self-regulating pushdown automaton without making any turn is exactly a common pushdown automaton. Therefore, $ALL\text{-}SPDA_0 = CF$. Next, we prove that one-turn all-move self-regulating pushdown automata are as powerful as Turing machines.

Theorem 4.3.1. $ALL\text{-}SPDA_1 = RE$.

The main idea of the proof is that every recursively enumerable language, L, can be expressed as $L = h(L(G) \cap L(H))$, where G and H are context-free grammars, and h is a homomorphism. Then, on the pushdown, automaton M simulates

1. G that generates a string, w, so that if a is on the top, M reads $h(a)$; then,
2. H that generates w, which is verified by R (no input is read).

Proof. For any recursively enumerable language, $L \subseteq \Delta^*$, there are context-free languages $L(G)$ and $L(H)$ and a homomorphism $h : \Sigma^* \to \Delta^*$ such that

$$L = h(L(G) \cap L(H))$$

(see Theorem 1.12 in [MS97]).

Suppose that $G = (N_G, \Sigma, P_G, S_G)$ and $H = (N_H, \Sigma, P_H, S_H)$ are context-free grammars in the Greibach normal form, i.e. all productions are of the form

$$A \to a\alpha\,,$$

where A is a nonterminal, a is a terminal, and α is a (possibly empty) string of nonterminals. Let us construct one-all-SPDA

$$M = (\{q_0, q, q_t, p, f\}, \Delta, \Sigma \cup N_G \cup N_H \cup \{Z\}, \delta, q_0, Z, \{f\}, R)\,,$$

where $Z \notin \Sigma \cup N_G \cup N_H$, with R made as follows:

1. add $(Zq_0 \to ZS_G q, Zq_t \to ZS_H p)$ to R
2. add $(Aq \to B_n \dots B_1 aq, Cp \to D_m \dots D_1 ap)$ to R if
   $A \to aB_1 \dots B_n \in P_G$ and
   $C \to aD_1 \dots D_m \in P_H$
3. add $(aqh(a) \to q, ap \to p)$ to R
4. add $(Zq \to Zq_t, Zp \to f)$ to R

Moreover, δ contains only the rules from the definition of R.

We prove that $w \in h(L(G) \cap L(H))$ if and only if $w \in L(M)$.

Only if Part: Let $w \in h(L(G) \cap L(H))$. There are $a_1, a_2, \dots, a_n \in \Sigma$ such that

$$a_1 a_2 \dots a_n \in L(G) \cap L(H)$$

and $w = h(a_1 a_2 \dots a_n)$, for some $n \in \mathbb{N}_0$. There are leftmost derivations

$$S_G \Rightarrow^n a_1 a_2 \dots a_n \text{ and } S_H \Rightarrow^n a_1 a_2 \dots a_n$$

of length n in G and H, respectively, because in every derivation step exactly one terminal symbol is derived. Thus, M accepts $h(a_1)h(a_2)\dots h(a_n)$ as

$$Zq_0 h(a_1)h(a_2)\dots h(a_n) \Rightarrow ZS_G qh(a_1)h(a_2)\dots h(a_n), \dots, Za_n qh(a_n) \Rightarrow Zq\,,$$

$$Zq \Rightarrow Zq_t\,,$$

$$Zq_t \Rightarrow ZS_H p, \dots, Za_n p \Rightarrow Zp, Zp \Rightarrow f\,.$$

In state q, by using its pushdown, M simulates G's derivation of $a_1 \dots a_n$ but reads $h(a_1)\dots h(a_n)$ as the input. In p, M simulates H's derivation of $a_1 a_2 \dots a_n$ but reads no input. As $a_1 a_2 \dots a_n$ can be derived in both G and H by making the same number of steps, the automaton can successfully complete the acceptance of w.

If Part: Notice that in one step, M can read only $h(a) \in \Delta^*$, for some $a \in \Sigma$. Let $w \in L(M)$, then $w = h(a_1)h(a_2)\dots h(a_n)$, for some $a_1, a_2, \dots, a_n \in \Sigma$. Consider M's acceptance of w

$$Zq_0 h(a_1)h(a_2)\dots h(a_n) \Rightarrow ZS_G qh(a_1)h(a_2)\dots h(a_n), \dots, Za_n qh(a_n) \Rightarrow Zq\,,$$

$$Zq \Rightarrow Zq_t\,,$$

$$Zq_t \Rightarrow ZS_H p, \dots, Za_n p \Rightarrow Zp, Zp \Rightarrow f\,.$$

As stated above, in q, M simulates G's derivation of $a_1 a_2 \dots a_n$, and then, in p, M simulates H's derivation of $a_1 a_2 \dots a_n$. It successfully completes the acceptance of w only if $a_1 a_2 \dots a_n$ can be derived in both G and H. Hence, the if part holds, too. $\square$

4.3.2 First-Move Self-Regulating Pushdown Automata

Although the fundamental results about self-regulating automata have been achieved in previous sections, there still remain several open problems concerning them. One of them is the question what is the language family accepted by n-turn first-move self-regulating pushdown automata, when $n \in \mathbb{N}$? It is clear that for $n = 0$ the language family accepted by zero-turn first-move self-regulating pushdown automata is exactly the family of all context-free languages.

4.3.3 Open Problems

Perhaps the most important open problems include 1 through 3 given next.

1. What is the language family accepted by n-turn first-move self-regulating pushdown automata, when $n \in \mathbb{N}$?
2. By analogy with standard deterministic finite and pushdown automata, introduce the deterministic versions of self-regulating automata. What is their power?
3. Discuss the closure properties under other language operations, such as the reversal.

Part II

Descriptional Complexity

5

Descriptional Complexity

This chapter studies descriptional complexity of partially parallel grammars and grammars regulated by context conditions, where the well-known results concerning this topic are supplemented and improved. The main aim of this chapter is to study how to describe partially parallel grammars and grammars regulated by context conditions in a reduced and succinct way with respect to the number of grammatical components, such as the number of nonterminals and special productions.

First of all, however, we define the notion of *descriptional complexity of grammars with respect to the number of nonterminals and special productions*.

Consider a family of languages, $\mathscr{L}$, and a family of grammars, $\mathscr{G}$, such that every language from $\mathscr{L}$ is generated by a grammar from $\mathscr{G}$, and every grammar from $\mathscr{G}$ generates only a language from $\mathscr{L}$, i.e. $L \in \mathscr{L}$ if and only if there is a grammar $G \in \mathscr{G}$ such that $L = L(G)$.

To reduce the number of nonterminals means to find a natural number (if it exists), k, such that for every language $L \in \mathscr{L}$, there is a grammar $G \in \mathscr{G}$ such that the set of all G's nonterminals, N, contains no more than k elements, i.e. $|N| \le k$, and G generates L, i.e. $L = L(G)$.

In other words, the question is what is the minimal k such that there is a subfamily, $\mathscr{H}$, of $\mathscr{G}$ consisting of grammars having no more than k nonterminals such that any language from $\mathscr{L}$ is generated by a grammar from $\mathscr{H}$.

The reduction of special productions is defined analogously, i.e., the aim is to find a natural number (if it exists), l, such that for every language $L \in \mathscr{L}$, there is a grammar $G \in \mathscr{G}$ with P being the set of all its productions, $P = P' \cup P''$, where P'' is the set of all special productions, such that $|P''| \le l$ and $L = L(G)$.

For instance, let P' be the set of all context-free and P'' the set of all remaining productions of P.

This chapter studies the simultaneous reduction of both the number of nonterminals and the number of special productions. In other words, in case of studied grammars, it is well-known that there are natural numbers k and l such that there is a subfamily, $\mathscr{H}$, of $\mathscr{G}$ having no more than k nonterminals and l special productions such that any language from $\mathscr{L}$ is generated by a grammar from $\mathscr{H}$. We decrease these numbers. More precisely, we prove that every recursively enumerable language is generated

(1) by a scattered context grammar with no more than four non-context-free productions and four nonterminals;
(2) by a multisequential grammar with no more than two selectors and two nonterminals;
(3) by a multicontinuous grammar with no more than two selectors and three nonterminals;
(4) by a context-conditional grammar of degree $(2,1)$ with no more than six conditional productions and seven nonterminals;
(5) by a simple context-conditional grammar of degree $(2,1)$ with no more than seven conditional productions and eight nonterminals;
(6) by a generalized forbidding grammar of degree two and index six with no more than ten conditional productions and nine nonterminals;
(7) by a generalized forbidding grammar of degree two and index four with no more than eleven conditional productions and ten nonterminals;
(8) by a generalized forbidding grammar of degree two and index nine with no more than eight conditional productions and ten nonterminals;
(9) by a generalized forbidding grammar of degree two and unlimited index with no more than nine conditional productions and eight nonterminals;
(10) by a semi-conditional grammar of degree $(2,1)$ with no more than seven conditional productions and eight nonterminals; and
(11) by a simple semi-conditional grammar of degree $(2,1)$ with no more than nine conditional productions and ten nonterminals.

5.1 Partially Parallel Grammars

This section studies descriptional complexity of partially parallel grammars. Specifically, descriptional complexity of scattered context grammars with respect to the number of nonterminals and context-sensitive productions, and descriptional complexity of multisequential and multicontinuous grammars with respect to the number of nonterminals and selectors.

5.1.1 Scattered Context Grammars

A scattered context grammar is an ordinary context-free grammar that uses its productions in a partially parallel way. More precisely, there is an integer n such that in each derivation step, no more than n nonterminals of the current sentential form is rewritten.

More details about scattered context grammars can be found in [Fer96, GW89, GH69, Mas07c, Med95, Med98b, Med02, MR71, Pău82, Vir73].

Definition 5.1.1. A *scattered context grammar*, G, is a quadruple

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

where

- N is a nonterminal alphabet,
- T is a terminal alphabet such that $N \cap T = \emptyset$,
- $S \in N$ is the start symbol, and
- P is a finite set of productions of the form

  $$(A_1, \ldots, A_n) \to (x_1, \ldots, x_n),$$

  for some $n \in \mathbb{N}$, where $A_i \in N$ and $x_i \in (N \cup T)^*$, for $i = 1, \ldots, n$. If $n \geq 2$, then the production is said to be *context-sensitive*; otherwise, the production is said to be *context-free*.

If $x = u_1 A_1 u_2 \ldots u_n A_n u_{n+1}$ and $y = u_1 x_1 u_2 \ldots u_n x_n u_{n+1}$, where $u_i \in (N \cup T)^*$, for all $i = 1, \ldots, n$, and

$$(A_1, \ldots, A_n) \to (x_1, \ldots, x_n) \in P,$$

then

$$x \Rightarrow y$$

in G. As usual, $\Rightarrow$ is extended to $\Rightarrow^i$, for $i \in \mathbb{N}_0$, $\Rightarrow^+$, and $\Rightarrow^*$. The language generated by a scattered context grammar, G, is defined as

$$L(G) = \{w \in T^* : S \Rightarrow^* w\}.$$

The last result concerning descriptional complexity of scattered context grammars is by Vaszil, who proved the following result (see [Vas05]). However, in spite of this fact, the presented construction of the proof is independently discovered by Masopust, Meduna, and Techet.[1]

Theorem 5.1.2. *Every recursively enumerable language is generated by a scattered context grammar with no more than five nonterminals and two context-sensitive productions.*

Proof. Let $L \subseteq T^*$ be any recursively enumerable language over an alphabet $T = \{a_1, \ldots, a_n\}$. Then, there is an extended Post correspondence problem[2]

$$E = (\{(u_1, v_1), \ldots, (u_r, v_r)\}, (z_{a_1}, \ldots, z_{a_n})),$$

[1] Unpublished result.

[2] Let $T = \{a_1, \ldots, a_n\}$, for some $n \geq 1$. An *extended Post correspondence problem* $E = (\{(u_1, v_1), \ldots, (u_r, v_r)\},\ (z_{a_1}, \ldots, z_{a_n}))$, where $u_i, v_i, z_{a_j} \in \{0,1\}^*$, for $i = 1, \ldots, r$, $j = 1, \ldots, n$. The language represented by E is the set $L(E) = \{b_1 \ldots b_k \in T^* : \exists s_1, \ldots, s_l \in \{1, \ldots, r\}, l \geq 1, v_{s_1} \ldots v_{s_l} = u_{s_1} \ldots u_{s_l} z_{b_1} \ldots z_{b_k}, k \geq 0\}$. It is well known that for each recursively enumerable language L, there is an extended Post correspondence problem, E, such that $L(E) = L$ (see Theorem 1 in [Gef88a]).

where $u_i, v_i, z_{a_j} \in \{0,1\}^*$, for $i = 1, \ldots, r,\ j = 1, \ldots, n$, such that $L(E) = L$. Define the scattered context grammar

$$G = (\{S, A, 0, 1, \$\}, T, P, S)$$

with P constructed as follows:

1. For every $a \in T$, add
   a) $(S) \rightarrow (z_a^R Sa)$, and
   b) $(S) \rightarrow (z_a^R Aa)$ to P;
2. a) For every $(u_i, v_i) \in E$, $1 \leq i \leq r$, add $(A) \rightarrow (u_i^R A v_i)$ to P;
   b) Add $(A) \rightarrow (\$\$)$ to P;
3. Add
   a) $(0, \$, \$, 0) \rightarrow (\$, \varepsilon, \varepsilon, \$)$,
   b) $(1, \$, \$, 1) \rightarrow (\$, \varepsilon, \varepsilon, \$)$, and
   c) $(\$) \rightarrow (\varepsilon)$ to P.

Examine the introduced productions to see that G generates $b_1 \ldots b_k \in L(E)$ by a derivation of this form:

$$\begin{aligned}
S &\Rightarrow z_{b_k}^R S b_k \\
&\Rightarrow z_{b_k}^R z_{b_{k-1}}^R S b_{k-1} b_k \\
&\Rightarrow^* z_{b_k}^R \ldots z_{b_2}^R S b_2 \ldots b_k \\
&\Rightarrow z_{b_k}^R \ldots z_{b_2}^R z_{b_1}^R A b_1 b_2 \ldots b_k \\
&\Rightarrow z_{b_k}^R \ldots z_{b_1}^R u_{s_l}^R A v_{s_l} b_1 \ldots b_k \\
&\Rightarrow^* z_{b_k}^R \ldots z_{b_1}^R u_{s_l}^R \ldots u_{s_1}^R A v_{s_1} \ldots v_{s_l} b_1 \ldots b_k \\
&\Rightarrow z_{b_k}^R \ldots z_{b_1}^R u_{s_l}^R \ldots u_{s_1}^R \$\$ v_{s_1} \ldots v_{s_l} b_1 \ldots b_k \\
&\quad = (u_{s_1} \ldots u_{s_l} z_{b_1} \ldots z_{b_k})^R \$\$ v_{s_1} \ldots v_{s_l} b_1 \ldots b_k \\
&\Rightarrow^* b_1 \ldots b_k.
\end{aligned}$$

Productions introduced in steps 1 and 2 of the construction find nondeterministically the solution of the extended Post correspondence problem which is subsequently verified by productions from step 3. Therefore $w \in L$ if and only if $w \in L(G)$ and the theorem holds. □

Now, we supplement this result as shown in the following theorem. Specifically, we prove that the number of nonterminals can be decreased, however, the number of conditional productions (nonsignificantly) increases.

Theorem 5.1.3. *Every recursively enumerable language is generated by a scattered context grammar with no more than four nonterminals and four context-sensitive productions.*

Basic idea.
The main idea of the proof and, actually, all proofs in this chapter is to simulate a terminal derivation of a grammar, G, in one of the Geffert normal forms.[3] To do this, we first apply all context-free productions as applied in the G's derivation, and then we simulate a non-context-free production, say $AB \to \varepsilon$, so that we mark with $'$ precisely one of As and one of Bs and check that these two marked symbols form a substring $A'B'$ of the current sentential form. If so, the marked symbols can be removed, which completes the simulation of the production $AB \to \varepsilon$ in G; otherwise, the derivation must be blocked. The formal proof follows.

Proof. Let $L \subseteq T^*$ be a recursively enumerable language and

$$G_2 = (\{S', A, B, C, D\}, T, P' \cup \{AB \to \varepsilon, CD \to \varepsilon\}, S')$$

be a grammar in the second Geffert normal form such that $L(G_2) = L$. Define the homomorphism $h : \{A, B, C, D\}^* \to \{0, 1\}^*$ so that $h(A) = h(B) = 00$, $h(C) = 10$, and $h(D) = 01$. Set $N = \{S, 0, 1, \$\}$. Define the scattered context grammar

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

with P constructed as follows:

1. $(S) \to (h(z)S1a1)$, where $S' \to zS'a \in P'$;
2. $(S) \to (h(u)Sh(v))$, where $S' \to uS'v \in P'$;
3. $(S) \to (11S)$;
4. $(S) \to (h(u)\$\$h(v))$, where $S' \to uv \in P'$;
5. $(\$) \to (\varepsilon)$;
6. $(0, 0, \$, \$, 0, 0) \to (\$, \varepsilon, \varepsilon, \varepsilon, \varepsilon, \$)$;
7. $(1, 0, \$, \$, 0, 1) \to (\$, \varepsilon, \varepsilon, \varepsilon, \varepsilon, \$)$;
8. $(1, 1, \$, \$, 1, 1) \to (11\$, \varepsilon, \varepsilon, \varepsilon, \varepsilon, \$)$;
9. $(1, 1, \$, \$, 1, 1) \to (\varepsilon, \varepsilon, \varepsilon, \varepsilon, \varepsilon, \varepsilon)$.

Consider a derivation of the form

$$S' \Rightarrow^* \alpha a_1 a_2 \ldots a_n \Rightarrow^* a_1 a_2 \ldots a_n,$$

where $\alpha \in \{A, B, C, D\}^*$, $a_i \in T$, for $i = 1, \ldots, n$, and neither $AB \to \varepsilon$ nor $CD \to \varepsilon$ has been applied in

$$S' \Rightarrow^* \alpha a_1 a_2 \ldots a_n.$$

Moreover, only productions $AB \to \varepsilon$ and $CD \to \varepsilon$ have been applied in

$$\alpha a_1 a_2 \ldots a_n \Rightarrow^* a_1 a_2 \ldots a_n.$$

If $a_1 a_2 \ldots a_n \neq \varepsilon$, then G can derive

$$S \Rightarrow^* 11h(\alpha)1a_1 11a_2 1 \ldots 1a_n 1$$

[3] See Definition 2.3.5 on page 9.

and, by productions constructed in 6 and 7, eliminate $h(\alpha)$. Thus,

$$S \Rightarrow^* 11\$\$1a_1 11a_2 1 \dots 1a_n 1 .$$

By productions constructed in 8 and 9, G eliminates all nonterminals 1 and \$.

If $a_1a_2\dots a_n = \varepsilon$, then G can derive $S \Rightarrow^* h(\alpha)$; then, by productions constructed in 6 and 7, G eliminates $h(\alpha)$. Thus, $S \Rightarrow^* \$\$$ in G. By the production constructed in 5, G eliminates both nonterminals \$. Therefore,

$$S' \Rightarrow^* a_1a_2\dots a_n \text{ implies } S \Rightarrow^* a_1a_2\dots a_n .$$

On the other hand, let

$$S \Rightarrow^* \alpha\$\$\beta \Rightarrow^* a_1a_2\dots a_n$$

be a derivation, where $\alpha \in \{00,01,11\}^*$, $\beta \in (\{00,01\} \cup \{1\}T\{1\})^*$, $a_i \in T$, for $i = 1,\dots,n$, and none of context-sensitive productions has been applied in $S \Rightarrow^* \alpha\$\$\beta$.

Notice that if a nonterminal occurs between the first and the second \$, then the nonterminal cannot be removed, so the derivation cannot generate a string of terminals.

If $a_1a_2\dots a_n = \varepsilon$, then $\beta \in \{00,01\}^*$, β does not contain 11 as a substring. Therefore, productions constructed in 8 and 9 cannot be applied in the derivation. Thus, neither can production 3 be applied, so α does not contain 11 as a substring, too. As the other productions simulate the productions from G_2, $S' \Rightarrow^* \varepsilon$ in G_2.

If $a_1a_2\dots a_n \neq \varepsilon$, then $\beta = \beta_1 1a_1 1\beta_2$, where $\beta_1 \in \{00,01\}^*$ and $\beta_2 \in (\{00,01\} \cup \{1\}T\{1\})^*$. After deleting β_1 by productions constructed in 6 and 7, the production constructed in 8 or 9 has to be applied. Therefore, $\alpha = \alpha_2 11\alpha_1$, where $\alpha_1 = \beta_1^R$ and $\alpha_2 \in \{0,1\}^*$. Thus,

$$S \Rightarrow^* \alpha\$\$\beta \Rightarrow^* \alpha_2 11\$\$1a_1 1\beta_2 .$$

We prove that $\alpha_2 = \varepsilon$ and $\beta_2 \in (\{1\}T\{1\})^*$ (by induction on $|\beta_2| \in \mathbb{N}_0$). At this point, the only productions that can be applied are productions constructed in 8 and 9. By applying the production constructed in 9, G makes

$$S \Rightarrow^* \alpha\$\$\beta \Rightarrow^* \alpha_2 11\$\$1a_1 1\beta_2 \Rightarrow \alpha_2 a_1 \beta_2 .$$

Therefore, $\alpha_2 a_1 \beta_2 \in T^*$ if and only if $\alpha_2 = \beta_2 = \varepsilon$. By applying the production constructed in 8, G makes

$$S \Rightarrow^* \alpha\$\$\beta \Rightarrow^* \alpha_2 11\$\$1a_1 1\beta_2 \Rightarrow \alpha_2 11\$a_1\$\beta_2 .$$

Therefore, if $\beta_2 = 00\beta_2'$, the prefix 00 can be removed only by the production constructed in 6. However, after using this production, the substring 11 attached to \$ appears between the two \$s, so it cannot be removed after that. The same is true for $\beta_2 = 01\beta_2''$. Thus, $\beta_2 = 1a_2 1\beta_3$. Then, by induction,

$$S \Rightarrow^* \alpha\$\$\beta = 11\gamma^R\$\$\gamma 1a_1 11a_2 1 \dots 1a_n 1 ,$$

where $\gamma \in \{00,01\}^*$. Since $h(A) = h(B) = 00$, $h(C) = 10$, and $h(D) = 01$, we get

$$S' \Rightarrow^* \delta_1\delta_2 a_1 a_2 \dots a_n \Rightarrow^* a_1 a_2 \dots a_n\,,$$

where $\delta_1 \in \{A,C\}^*$, $\delta_2 \in \{B,D\}^*$, $h(\delta_1) = \gamma^R$, and $h(\delta_2) = \gamma$.
Hence, the theorem holds. $\square$

5.1.2 Propagating Scattered Context Grammars

Originally, in 1969 Greibach and Hopcroft introduced scattered context grammars without ε-productions (see [GH69]). In these days, such scattered context grammars are called propagating scattered context grammars. Although a lot of problems concerning the closure properties of propagating scattered context grammars have been solved since their introduction, there are still some problems open. It is not hard to prove that all propagating scattered context grammars generate languages which are context sensitive. The proof of this assertion is based on the so-called workspace theorem (see, for example, Theorem III.10.1 in [Sal73]). One of the most famous open problems concerning propagating scattered context grammars is the question whether there is a context sensitive language which is not scattered context.

In this section, we give some nontrivial examples of languages generating by propagating scattered context grammars.

First of all, recall that a construction (without a proof) of a scattered context grammar generating the language

$$\{a^{2^{n+1}+n+1} : n \geq 1\}$$

is presented in [ER79]. Based on this construction, a proof that the language

$$\{a^{2^n} : n \geq 0\}$$

is a scattered context language is given in [Mas07c].

Definition 5.1.4. A scattered context grammar $G = (N,T,P,S)$ is said to be a *propagating scattered context grammar* if any production in P is of the form

$$(A_1,\dots,A_n) \to (x_1,\dots,x_n)\,,$$

for some $n \geq 1$, where $A_i \in N$ and $x_i \in V^+$, for all $i = 1,\dots,n$.

We now give some nontrivial examples of languages generated by a propagating scattered context grammar.

Theorem 5.1.5. *For any integer $k \geq 2$, there is a propagating scattered context grammar G with six nonterminals and eight productions such that*

$$L(G) = \{a^{k^n} : n \geq 0\}\,.$$

Proof. Define the following propagating scattered context grammar

$$G = (\{S, A, A', X, Y, Z\}, \{a\}, P, S),$$

where P contains the following productions:

1. $(S) \to (a)$
2. $(S) \to (a^k)$
3. $(S) \to (A'A^{k-1}XY)$
4. $(A', A, X, Y) \to (a^{k-1}, A', X, A^kY)$
5. $(A', X, Y) \to (a^{k-1}, A', A^{k-1}XY)$
6. $(A', X, Y) \to (Z, Z, a^{k-1})$
7. $(Z, A, Z) \to (Z, a^{k-1}, Z)$
8. $(Z, Z) \to (a, a^{k-1})$

Consider a string of the form

$$a^*A'A^nXA^mY,$$

where $m, n \geq 0$. By production 6,

$$a^*A'A^nXA^mY \Rightarrow a^*ZA^nZA^ma^{k-1} \quad [6].$$

Then, we can either finish the derivation by production 8 (if $n = 0$), or continue by production 7 as long as possible, followed by production 8;

$$a^*ZA^nZA^ma^{k-1} \Rightarrow^* a^*Za^{n(k-1)}ZA^ma^{k-1} \Rightarrow a^*aa^{n(k-1)}aA^ma^{k-1} \quad [7^*8].$$

It is easy to see that

$$a^*aa^{n(k-1)}aA^ma^{k-1} \in \mathscr{L}(G) \text{ if and only if } m = 0.$$

Therefore, to apply production 6, the sentential form must be of the form $a^*A'A^nXY$, for some $n \geq 1$ (from the previous and production 3).

If production 5 is applied to $a^*A'A^nXA^mY$,

$$a^*A'A^nXA^mY \Rightarrow a^*a^{k-1}A^nA'A^mA^{k-1}XY \quad [5],$$

then the derivation is blocked; indeed, A on the left-hand side of A' cannot be removed. Thus, if $m \geq 1$, the only possible derivation is to apply production 4 as long as there is A between A' and X.

From the previous follows that if the first production applied to

$$a^{k^n-k}A'A^{k^n-1}XY$$

is production 4, then the derivation continues by production 4 as long as there is A between A' and X, followed by one application of production 5,

$$\begin{aligned} a^{k^n-k}A'A^{k^n-1}XY &\Rightarrow^+ a^{k^n-k}a^{(k-1)(k^n-1)}A'XA^{k(k^n-1)}Y \quad [44^*] \\ &= a^{k^{(n+1)}-2k+1}A'XA^{k^{(n+1)}-k}Y \\ &\Rightarrow a^{k^{(n+1)}-2k+1}a^{k-1}A'A^{k^{(n+1)}-k}A^{k-1}XY \quad [5], \\ &= a^{k^{(n+1)}-k}A'A^{k^{(n+1)}-1}XY \end{aligned}$$

where $n \geq 1$.

Now, the derivation can successfully rewrite all A's to a^{k-1} by productions 6, 7, and 8, i.e.

$$a^{k^n-k}A'A^{k^n-1}XY \Rightarrow^* a^{k^n-k}aa^{(k-1)(k^n-1)}a^{k-1}a^{k-1} = a^{k^{n+1}} \quad [67^+8],$$

or continue by production 4. Then, it follows by induction.

Finally, we summarize all possible terminal derivations.

$$\begin{array}{lll} S \Rightarrow & a & [1] \\ S \Rightarrow & a^k & [2] \\ S \Rightarrow & A'A^{k-1}XY & [3] \\ \Rightarrow^* & a^{k^2-1}A'A^{k^2-1}XY & [44^*5] \\ \vdots & & \\ \Rightarrow^* & a^{k^{n-1}-k}A'A^{k^{n-1}-1}XY & [44^*5] \\ \Rightarrow^* & a^{k^n} & [67^+8] \end{array}$$

Thus, any derivation generating a terminal string, w, is in one of the following three forms:

$$S \Rightarrow^* w\ [1] \quad \text{or} \quad S \Rightarrow^* w\ [2] \quad \text{or} \quad S \Rightarrow^* w\ [3(44^*5)^*67^+8].$$

□

Based on the previous theorem, the following theorem can be proved.

Theorem 5.1.6. *For any integers $k,l \geq 2$, there is a propagating scattered context grammar G with twelve nonterminals and fourteen productions such that*

$$L(G) = \{a^{l^{k^n}} : n \geq 0\}.$$

Proof. Define the following propagating scattered context grammar

$$G = (\{S,A,A',A'',B,C,X,X_2,X_3,Y,Z,Z'\},\{a\},P,S),$$

where P contains the following productions:

1. $(S) \rightarrow (a^l)$
2. $(S) \rightarrow \left(a^{l^k}\right)$
3. $(S) \rightarrow \left(a^{l^{k^2}}\right)$

4. $(S) \to (A''A^{l-1}X_2B^{k^2-3}A'C^{k^2-1}XY)$
* first stage
5. $(A',C,X,Y) \to (B^{k-1},A',X,C^kY)$
6. $(A',X,Y) \to (B^{k-1},A',C^{k-1}XY)$
7. $(A',X,Y) \to (Z,Z,Y)$
8. $(Z,C,Z,Y) \to (Z,B^{k-1},Z,Y)$
9. $(Z,Z,Y) \to (B,B^{k-1},X_3)$
* second stage
10. $(A'',A,X_2,X_3) \to (a^{l-1},A'',X_2A^l,X_3)$
11. $(A'',X_2,B,X_3) \to (a^{l-1},A'',A^{l-1}X_2,X_3)$
12. $(A'',X_2,X_3) \to (Z',Z',X_3)$
13. $(Z',A,Z',X_3) \to (Z',a^{l-1},Z',X_3)$
14. $(Z',Z',X_3) \to (a,a^{l-1},a^{l-1})$

First, notice that if the production

$$(S) \to (A''A^{l-1}X_2B^{k^2-3}A'C^{k^2-1}XY)$$

is applied, then none of the productions of the second stage is applicable; clearly, there is no X_3 in the derivation, and X_3 does not appear in any sentential form of the derivation all the time productions from the first stage are applicable. Notice also that the productions of the first stage are very similar to the productions of the grammar constructed in the proof of Theorem 5.1.5. Therefore, it is not hard to see that, by the productions from the first stage,

$$B^{k^n-3}A'C^{k^n-1}XY \Rightarrow^* B^{k^{n+1}-2}X_3\,,$$

for $n \geq 2$. Thus, we have the following derivation;

$$\begin{aligned} S \;&\Rightarrow\; A''A^{l-1}X_2B^{k^2-3}A'C^{k^2-1}XY \\ &\Rightarrow^*\; A''A^{l-1}X_2B^{k^n-2}X_3\,, \end{aligned}$$

for $n \geq 3$. Consider a string of the form

$$a^*A''A^pX_2A^qB^rX_3\,,$$

where $p,q,r \geq 0$. By production 12,

$$a^*A''A^pX_2A^qB^rX_3 \Rightarrow a^*Z'A^pZ'A^qB^rX_3 \quad [12]\,.$$

Then, we can either finish the derivation by production 14 (if $p = 0$), or continue by production 13 as long as possible, followed by production 14;

$$\begin{aligned} a^*Z'A^pZ'A^qB^rX_3 \;&\Rightarrow^*\; a^*Z'a^{p(l-1)}Z'A^qB^rX_3 && [(13)^+] \\ &\Rightarrow\; a^*aa^{p(l-1)}a^{l-1}A^qB^ra^{l-1} && [(14)]\,. \end{aligned}$$

It is easy to see that

$$a^*aa^{p(l-1)}a^{l-1}A^qB^ra^{l-1} \in \mathscr{L}(G) \text{ if and only if } q = r = 0\,.$$

Therefore, to apply production 12, the sentential form must be of the form

$$a^*A''A^pX_2X_3\,,$$

for some $p \geq 1$.

If production 11 is applied to $a^*A''A^pX_2A^qB^rX_3$,

$$a^*A''A^pX_2A^qB^rX_3 \Rightarrow a^*a^{l-1}A^pA''A^qA^{l-1}X_2B^{r-1}X_3 \quad [11]\,,$$

then the derivation is blocked; indeed, A on the left-hand side of A'' cannot be removed. Thus, if $r \geq 1$, then the only possible derivation is to apply production 10 as long as there is A between A'' and X_2.

We prove that all sentential forms of a terminal derivation containing X_3, i.e. of the second stage, are of the form

$$a^{l^{m-1}-l}A''A^{l^{m-1}-1}X_2B^{k^n-m}X_3\,,$$

for all $m = 2,\ldots,k^n$, $n \geq 3$.

For $m = 2$, we have

$$A''A^{l-1}X_2B^{k^n-2}X_3\,,$$

and this is the start sentential form of the second stage. For $m = 2^n$, we have

$$a^{l^{k^n-1}-l}A''A^{l^{k^n-1}-1}X_2X_3\,,$$

and, as proved above,

$$a^{l^{k^n-1}-l}A''A^{l^{k^n-1}-1}X_2X_3 \Rightarrow^* a^{l^{k^n-1}-l}aa^{(l-1)(l^{k^n-1}-1)}a^{l-1}a^{l-1} = a^{l^{k^n}}\,.$$

Thus, assume that $2 \leq m < k^n$. Then,

$$\begin{aligned}
& a^{l^{m-1}-l}A''A^{l^{m-1}-1}X_2B^{k^n-m}X_3 \\
\Rightarrow^*\ & a^{l^{m-1}-l}a^{(l-1)(l^{m-1}-1)}A''X_2A^{l(l^{m-1}-1)}B^{k^n-m}X_3 \ [10] \\
=\ & a^{l^m-2l+1}A''X_2A^{l^m-l}B^{k^n-m}X_3 \\
\Rightarrow\ & a^{l^m-2l+1}a^{l-1}A''A^{l^m-l}A^{l-1}X_2B^{k^n-m-1}X_3 \ [11] \\
=\ & a^{l^m-l}A''A^{l^m-1}X_2B^{k^n-(m+1)}X_3\,.
\end{aligned}$$

Finally, we summarize all possible terminal derivations.

$$\begin{array}{lll}
S \Rightarrow & a^l & [1] \\
S \Rightarrow & a^{l^k} & [2] \\
S \Rightarrow & a^{l^{k^2}} & [3] \\
S \Rightarrow^* & A''A^{l-1}X_2B^{k^n-2}X_3 & [4(55^*6)^*78^+9] \\
\Rightarrow^* & a^{l^{k^n}} & [((10)^+(11))^*(12)(13)^+(14)]\,.
\end{array}$$

$\square$

Open Problem

Another interesting open problem (a subproblem of the most famous open problem mentioned above) is whether there is a propagating scattered context grammar generating the set of all prime numbers, i.e. is there a propagating scattered context grammar G such that

$$L(G) = \{a^p : p \text{ is a prime number}\}\,?$$

Complement

Finally, we can prove that if propagating scattered context grammars are closed under complement, then they can generate the set of all prime numbers. To prove this, we give a construction of a propagating scattered context grammar generating all composite numbers. Before doing so, note that the question whether propagating scattered context grammars are closed under complement or not is the next famous open problem.

Theorem 5.1.7. *There is a propagating scattered context grammar G such that*

$$L(G) = \{a^k : k = mn, m, n \geq 2\}\,.$$

Proof. Define the following propagating scattered context grammar

$$G = (\{S, S_1, S_2, A, B, X, Y, Z\}, \{a\}, P, S)\,,$$

where P contains the following productions:

1. $(S) \to (a)$
2. $(S) \to (a^4)$
3. $(S) \to (S_1S_2)$
4. $(S_1) \to (S_1A)$
5. $(S_1) \to (XA)$
6. $(S_2) \to (S_2B)$
7. $(S_2) \to (YB)$
8. $(X, A, Y) \to (a, X, YA)$
9. $(X, Y, B) \to (a^2, X, Y)$
10. $(X, Y, B) \to (a^2, X, Z)$
11. $(X, A, Z) \to (X, a, Z)$
12. $(X, Z) \to (a, a)$

It is easy to see that

$$S \Rightarrow^* XA^{n-2}YB^{m-1}$$

by productions 3, 4, 5, 6, 7, for some $m \geq 2$ and $n \geq 3$. Then,

$$\begin{array}{rlll} XA^{n-2}YB^{m-1} & \Rightarrow^* & a^{n-2}XYA^{n-2}B^{m-1} & [8^+] \\ & \Rightarrow & a^{n-2}a^2XA^{n-2}YB^{m-2} & [9] \\ & \Rightarrow^* & a^n a^n XA^{n-2}YB^{m-3} & [8^+9] \\ & \Rightarrow^* & a^{n(m-2)}a^{n-2}XYA^{n-2}B & [(8^+9)^*8^+] \\ & \Rightarrow & a^{n(m-1)}XA^{n-2}Z & [10] \\ & \Rightarrow^* & a^{n(m-1)}a^n & [(11)^*(12)] \\ & = & a^{nm}\,. & \end{array}$$

The proof is left to the reader. □

5.1.3 Multisequential Grammars

A multisequential grammar is a context-free grammar, where also terminal symbols can be rewritten. In addition, these grammars have a mechanism that chooses symbols of the current sentential form that are supposed to be rewritten. Such mechanisms are called *selectors*. Then, during any derivation step, all chosen symbols are rewritten.

Definition 5.1.8. A *multisequential grammar*, G, is a quintuple

$$G = (N, T, P, S, K)\,,$$

where

- N is a nonterminal alphabet,
- T is a terminal alphabet such that $N \cap T = \emptyset$,
- $S \in N$ is the start symbol,
- P is a finite set of productions of the form

  $$a \to x\,,$$

  where $a \in V = N \cup T$ and $x \in V^*$, and
- K is a finite set of selectors of the form

  $$X_1\mathbf{act}(Y_1)X_2 \dots X_n\mathbf{act}(Y_n)X_{n+1}\,,$$

  where n is a positive integer,
  - $X_i \in \{Z^* : Z \subseteq V\}$, for $i = 1, \dots, n+1$, and
  - $Y_j \in \{Z : Z \subseteq V, Z \neq \emptyset\}$, for $j = 1, \dots, n$.

If $x = u_1a_1u_2a_2u_3 \dots u_na_nu_{n+1}$, $y = u_1x_1u_2x_2u_3 \dots u_nx_nu_{n+1}$, and K contains a selector

$$X_1\mathbf{act}(Y_1)X_2 \dots X_n\mathbf{act}(Y_n)X_{n+1}$$

satisfying $u_i \in X_i$, for $i = 1, \dots, n+1$, $a_j \in Y_j$, and $a_j \to x_j \in P$, for $j = 1, \dots, n$, then

$$x \Rightarrow y$$

in G. As usual, $\Rightarrow$ is extended to $\Rightarrow^i$, for $i \in \mathbb{N}_0$, $\Rightarrow^+$, and $\Rightarrow^*$. The language generated by a multisequential grammar, G, is defined as

$$L(G) = \{w \in T^* : S \Rightarrow^* w\}\,.$$

In [Med97c], the following result is proved.

Theorem 5.1.9. *Every recursively enumerable language is generated by a multisequential grammar with no more than six nonterminals.*

Here, we improve this result. Specifically, we prove that no more than two nonterminals are needed and, moreover, we give a limit to the number of selectors. First, however, we prove the following auxiliary lemma.

Lemma 5.1.10. *Every recursively enumerable language is generated by a multisequential grammar with no more than three nonterminals and two selectors.*

Proof. Let $L \subseteq T^*$ be a recursively enumerable language and

$$G_3 = (\{S, A, B\}, T, P \cup \{ABBBA \to \varepsilon\}, S)$$

be a grammar in the third Geffert normal form such that $L(G_3) = L$. Define the multisequential grammar

$$G = (\{S, A, B\}, T, P \cup \{A \to \varepsilon,\ B \to \varepsilon\}, S, K)$$

with K containing these two selectors:

1. $\{A, B\}^* \mathbf{act}(S)(\{A, B\} \cup T)^*$,
2. $\{A, B\}^* \mathbf{act}(A)\mathbf{act}(B)\mathbf{act}(B)\mathbf{act}(B)\mathbf{act}(A)(\{A, B\} \cup T)^*$.

Observe that $L(G) = L(G_3)$. □

Now, based on the previous lemma, the main result of this part can be proved.

Theorem 5.1.11. *Every recursively enumerable language is generated by a multisequential grammar with no more than two nonterminals and two selectors.*

Proof. Consider G constructed in the proof of Lemma 5.1.10. Define the homomorphism

$$h : (\{S, A, B\} \cup T)^* \to (\{S, A\} \cup T)^*$$

as $h(b) = b$, for $b \in T$, $h(S) = S$, $h(A) = aAa$, and $h(B) = aAAa$, where $a \in T$ is a terminal symbol. Define the multisequential grammar

$$G' = (\{S, A\}, T, \{S \to h(\alpha) : S \to \alpha \in P\} \cup \{A \to \varepsilon, a \to \varepsilon\}, S, K)$$

with K containing these two selectors:

1. $\{A, a\}^* \mathbf{act}(S)(\{A\} \cup T)^*$,
2. $\{A, a\}^* \mathbf{act}(a)\mathbf{act}(A)\mathbf{act}(a)\mathbf{act}(a)\mathbf{act}(A)\mathbf{act}(A)\mathbf{act}(a)$
   $\mathbf{act}(a)\mathbf{act}(A)\mathbf{act}(A)\mathbf{act}(a)\mathbf{act}(a)\mathbf{act}(A)\mathbf{act}(A)\mathbf{act}(a)$
   $\mathbf{act}(a)\mathbf{act}(A)\mathbf{act}(a)(\{A\} \cup T)^*$.

Observe that $S \to \alpha$ is a production in G if and only if $S \to h(\alpha)$ is a production in G'. If

$$uABBBAv \Rightarrow uv$$

in G, where $u \in \{A,B\}^*$ and $v \in \{A,B\}^*T^*$, then

$$h(u)aAaaAAaaAAaaAAaaAah(v) \Rightarrow h(uv)$$

in G' (by selector 2), and vice versa.

Hence, the theorem holds. $\square$

5.1.4 Multicontinuous Grammars

A multicontinuous grammar is a multisequential grammar differing in the following detail—the whole nonempty strings of symbols, not only one symbol, is chosen to be rewritten. Then, in any derivation step, all these symbols are rewritten.

Definition 5.1.12. A *multicontinuous grammar*, G, is a quintuple

$$G = (N,T,P,S,K),$$

where

- N is a nonterminal alphabet,
- T is a terminal alphabet such that $N \cap T = \emptyset$,
- $S \in N$ is the start symbol,
- P is a finite set of productions of the form

$$a \to x,$$

  where $a \in V = N \cup T$ and $x \in V^*$, and
- K is a finite set of selectors of the form

$$X_1\mathbf{act}(Y_1)X_2 \dots X_n\mathbf{act}(Y_n)X_{n+1},$$

  where n is a positive integer,
  - $X_i \in \{Z^* : Z \subseteq V\}$, for $i = 1,\dots,n+1$, and
  - $Y_j \in \{Z^+ : Z \subseteq V, Z \neq \emptyset\}$, for $j = 1,\dots,n$.

For every $v \in V^+$, where $v = a_1 \dots a_{|v|}$ with $a_i \in V$, for $i = 1,\dots,|v|$, define the language

$$ContinuousRewriting(v) \subseteq V^*$$

by this equivalence: for every $z \in V^*$, $z \in ContinuousRewriting(v)$ if and only if $a_i \to x_i \in P$, for $i = 1,\dots,|v|$, and $z = x_1 \dots x_{|v|}$.

If $x = u_1y_1u_2y_2u_3 \dots u_ny_nu_{n+1}$, $y = u_1z_1u_2z_2u_3 \dots u_nz_nu_{n+1}$, and K contains a selector

$$X_1\mathbf{act}(Y_1)X_2 \dots X_n\mathbf{act}(Y_n)X_{n+1}$$

such that $u_i \in X_i$, for $i = 1,\dots,n+1$, $y_j \in Y_j$, and

$$z_j \in ContinuousRewriting(y_j)\,,$$

for $j = 1, \dots, n$, then

$$x \Rightarrow y$$

in G. As usual, $\Rightarrow$ is extended to $\Rightarrow^i$, for $i \geq 0$, $\Rightarrow^+$, and $\Rightarrow^*$. The language generated by a multicontinuous grammar, G, is defined as

$$L(G) = \{w \in T^* : S \Rightarrow^* w\}\,.$$

In [Med98a], the following result is proved.

Theorem 5.1.13. *Every recursively enumerable language is generated by a multicontinuous grammar with no more than six nonterminals.*

We improve this result in the following way. Again, we give a limit to the number of selectors.

Theorem 5.1.14. *Every recursively enumerable language is generated by a multicontinuous grammar with no more than three nonterminals and two selectors.*

Proof. Let $L \subseteq T^*$ be a recursively enumerable language. Let

$$G_3 = (\{S, A, B\}, T, P \cup \{ABBBA \rightarrow \varepsilon\}, S)$$

be a grammar in the third Geffert normal form such that $L(G_3) = L$. Define the homomorphism

$$h : (\{S, A, B\} \cup T)^* \rightarrow (\{S, (,)\} \cup T)^*$$

as $h(a) = a$, for $a \in T$, $h(S) = S$, $h(A) = ()$, and $h(B) = (b)$, where $b \in T$ is a terminal symbol. Define the multicontinuous grammar

$$G = (\{S, (,)\}, T, \{S \rightarrow h(\alpha) : S \rightarrow \alpha \in P\} \cup \{(\rightarrow \varepsilon,) \rightarrow \varepsilon, b \rightarrow \varepsilon\}, S, K)$$

with K containing these two selectors:

1. $\{(,), b\}^* \mathbf{act}(S^+)(\{(,)\} \cup T)^*$,
2. $\{(,), b\}^* \mathbf{act}((^+)\mathbf{act}()^+)\mathbf{act}((^+)\mathbf{act}(b^+)\mathbf{act}()^+)$
   $\mathbf{act}((^+)\mathbf{act}(b^+)\mathbf{act}()^+)\mathbf{act}((^+)\mathbf{act}(b^+)\mathbf{act}()^+)$
   $\mathbf{act}((^+)\mathbf{act}()^+)(\{(,)\} \cup T)^*$.

At the beginning of any derivation, only selector 1 is applicable. After eliminating S, the other selector is applicable. Moreover, as there is no more than one substring of the form

$$h(ABBBA) = ()(b)(b)(b)()$$

in each sentential form (see [Gef88a]), selector 2 is applicable only on no more than one substring. As there is no occurrence of substrings ((or)) in any sentential form, this theorem holds. □

5.2 Context-Conditional Grammars

A context-conditional grammar is an ordinary context-free grammar, where a set of permitting and a set of forbidding contexts are associated with each production. Then, a production is applicable if and only if it is applicable as a context-free production and each permitting context and no forbidding context associated with this production is a substring of the current sentential form.

Definition 5.2.1. A *context-conditional grammar*, G, is a quadruple

$$G = (N, T, P, S)\,,$$

where

- N is a nonterminal alphabet,
- T is a terminal alphabet such that $N \cap T = \emptyset$,
- $S \in N$ is the start symbol, and
- P is a finite set of productions of the form

  $$(X \to \alpha, Per, For)\,,$$

  where $X \in N$, $\alpha \in (N \cup T)^*$, and Per, $For \subseteq (N \cup T)^+$ are finite sets. If $Per \cup For \neq \emptyset$, then the production is said to be *conditional*.

G has *degree* (i, j) if for all productions $(X \to \alpha, Per, For) \in P$,

$$\max(Per)^4 \leq i$$

and

$$\max(For) \leq j\,.$$

G has *index* k if

$$\max\{|Per| + |For| : (X \to \alpha, Per, For) \in P\} \leq k\,.$$

For $x_1, x_2 \in (N \cup T)^*$, $x_1 X x_2$ *directly derives* $x_1 \alpha x_2$ according to the production $(X \to \alpha, Per, For) \in P$, denoted by $x_1 X x_2 \Rightarrow x_1 \alpha x_2$, if

$$Per \subseteq sub(x)$$

and

$$For \cap sub(x) = \emptyset\,.$$

As usual, $\Rightarrow$ is extended to $\Rightarrow^i$, for $i \in \mathbb{N}_0$, $\Rightarrow^+$, and $\Rightarrow^*$. The language generated by a context-conditional grammar, G, is defined as

$$L(G) = \{w \in T^* : S \Rightarrow^* w\}\,.$$

[4] See Definition 2.2.10 on page 7.

In [MŠ05], a proof that context-conditional grammars generate the whole family of recursively enumerable languages is given. However, descriptional complexity of context-conditional grammars has not been studied so far. Next, a result concerning this topic follows.

Theorem 5.2.2. *Every recursively enumerable language is generated by a context-conditional grammar of degree* $(2,1)$ *and index two with no more than six conditional productions and seven nonterminals.*

Proof. Let L be a recursively enumerable language. Then, there is a grammar

$$G_1 = (\{S,A,B,C\},T,P \cup \{ABC \to \varepsilon\},S)$$

in the first Geffert normal form such that $L = L(G_1)$. Construct the grammar

$$G = (\{S,A,B,C,A',B',C'\},T,P' \cup P'',S)\,,$$

where

$$P' = \{(X \to \alpha,\emptyset,\emptyset) : X \to \alpha \in P\}\,,$$

and P'' contains the following six conditional productions:

1. $(A \to A',\emptyset,\{A',C'\})$,
2. $(B \to B',\{A'\},\{B'\})$,
3. $(C \to C',\{A'B'\},\{C'\})$,
4. $(A' \to \varepsilon,\{B'C'\},\emptyset)$,
5. $(B' \to \varepsilon,\{C'\},\{A'\})$,
6. $(C' \to \varepsilon,\emptyset,\{A',B'\})$.

To prove that $L(G_1) \subseteq L(G)$, consider a derivation

$$S \Rightarrow^* wABCw'v \Rightarrow ww'v$$

in G_1 by productions from P and the only one production $ABC \to \varepsilon$, where $w,w' \in \{A,B,C\}^*$ and $v \in T^*$. Then, $S \Rightarrow^* wABCw'v$ in G by productions from P'. By productions 1, 2, 3, 4, 5, and 6,

$$\begin{aligned} wABCw'v &\Rightarrow wA'BCw'v \\ &\Rightarrow wA'B'Cw'v \\ &\Rightarrow wA'B'C'w'v \\ &\Rightarrow wB'C'w'v \\ &\Rightarrow wC'w'v \\ &\Rightarrow ww'v. \end{aligned}$$

The inclusion follows by induction.

To prove that $L(G_1) \supseteq L(G)$, consider a terminal derivation. We prove that, after eliminating S, in each six consecutive steps, G can do nothing else than to remove a substring ABC. To prove it, notice first that to remove A' or B', i.e. A or B, C' has to be in the sentential form (see productions 4 and 5). However, to obtain C' to the sentential form, production 3 has to be applied. Then, $A'B'$ has to be a substring of a former sentential form. Thus, productions 1 and 2 had to be applied before and in this order. It is also easy to see, according to the forbidding contexts of productions 1, 2, and 3, that there cannot be more than one occurrence of nonterminals A', B', and C' in any sentential form. Therefore, according to the permitting contexts of productions 3 and 4, $A'B'C'$ is a substring of the current sentential form, and, moreover, there cannot be a terminal between any two nonterminals. The derivation is of the form $S \Rightarrow^* w_1w_2w_3$ in G by productions from P', where $w_1 \in \{A,AB\}^*$, $w_2 \in \{BC,C\}^*$, and $w_3 \in T^*$, and $w_1w_2w_3 \Rightarrow^* w_3$. Then, $S \Rightarrow^* w_1w_2w_3$ in G_1 by productions from P. We prove that $w_1w_2w_3 \Rightarrow^* w_3$ in G_1.

For $w_1w_2 = \varepsilon$, the proof is done. For $w_1w_2 \neq \varepsilon$, $w_1w_2 = wABCw'$, where $w \in \{A,AB\}^*$ and $w' \in \{BC,C\}^*$. Thus, at the beginning, only production 1, then 2, and then 3 is applicable. Then, only production 4 is applicable, and, after that, only production 5 is applicable. Finally, production 6 can be applied;

$$\begin{aligned} wABCw'w_3 &\Rightarrow^3 wA'B'C'w'w_3 \\ &\Rightarrow wB'C'w'w_3 \\ &\Rightarrow wC'w'w_3 \\ &\Rightarrow ww'w_3. \end{aligned}$$

Thus, if $S \Rightarrow^* w_1w_2w_3 \Rightarrow^* w_3$ in G, where $w_1 \in \{A,AB\}^*$, $w_2 \in \{BC,C\}^*$, and $w_3 \in T^*$, then $S \Rightarrow^* w_1w_2w_3 \Rightarrow^* w_3$ in G_1. □

To complete this section, note that it is proved (see [May72], [DP89], and [Sal73] for a complete proof) that every recursively enumerable language is generated by a context-conditional grammar of degree $(1,1)$. However, in this case, the number of nonterminals and conditional productions is not limited.

Note that context-conditional grammars of degree $(1,0)$, $(0,1)$, and $(1,1)$, originally introduced by van der Walt in [vdW70], are known as *random context grammars* (or *permitting grammars*), *forbidding grammars*, and *random context grammars with appearance checking*, respectively. It is also known that random context grammars are as powerful as type 0 grammars, and that neither permitting nor forbidding grammars are as powerful as type 0 grammars (they do not even generate all recursive languages).[5] Finally, the relation between language families generated by these two types of grammars is not known. For all these results in more details see [BF96] (or [RS97, pages 136 and 137] for an overview).

5.2.1 Context-Conditional Grammars with Linear and

[5] In [MŠ05], authors mention that the family of languages generated by forbidding grammars is strictly included in the family of context-sensitive languages. However, the proof shown there is not correct. Nevertheless, this is true for forbidding grammars without erasing productions, i.e. without productions of the form $A \to \varepsilon$.

Regular Productions

This section proves that context-conditional grammars having only linear (regular) productions[6] characterize exactly the family of all linear (regular) languages. Thus, this section proves that linear productions do not increase the generative power of such grammars, whereas context-free productions do.

Theorem 5.2.3. *Context-conditional grammars having only linear productions are as powerful as linear grammars.*

Proof. Any linear grammar[7] can be seen as a context-conditional grammar with empty permitting and forbidding contexts.

To prove the other direction, let $G = (N, T, P, S)$ be a context-conditional grammar with linear productions. Without loss of generality, we can assume that S does not occur on the right-hand side of any production. Set

$$\mathbf{PER} = \bigcup_{(X\to\alpha,Per,For)\in P} Per\,, \quad \mathbf{FOR} = \bigcup_{(X\to\alpha,Per,For)\in P} For\,.$$

We can assume that $S \notin \mathbf{PER} \cup \mathbf{FOR}$ because S occurs in a sentential form only at the very beginning of the derivation and never more. If, moreover, $S \in For$, for some $(S \to \alpha, Per, For) \in P$, then this production is not applicable and can be removed from P without modifying the generated language. Analogously for $S \in Per$, for some $(X \to \alpha, Per, For) \in P$, $X \neq S$.

Let m denote the length of the longest string from $\mathbf{PER} \cup \mathbf{FOR}$, and let

$$head(a_1a_2\ldots a_k) \;=\; \begin{cases} a_1a_2\ldots a_m & \text{for } m \leq k \\ a_1a_2\ldots a_k & \text{otherwise,} \end{cases}$$

and

$$tail(a_1a_2\ldots a_k) \;=\; \begin{cases} a_{k-m+1}a_{k-m+2}\ldots a_k & \text{for } m \leq k \\ a_1a_2\ldots a_k & \text{otherwise.} \end{cases}$$

Construct the following linear grammar $\bar{G} = (\bar{N}, T, \bar{P}, [S, \emptyset, \emptyset, \varepsilon, \varepsilon])$, where

$$\bar{N} = \{[X, P, F, u, v] : X \in N, P \subseteq \mathbf{PER}, F \subseteq \mathbf{FOR},$$
$$u, v \in T^*, |u|, |v| \leq m\}\,,$$

and $\bar{P}$ is constructed as follows.

1. For all productions $(X \to aY, \{u_1, \ldots, u_k\}, \{v_1, \ldots, v_l\})$, add

$$[X, P, F, u, v] \to a[Y, P', F', tail(ua), v]$$

into $\bar{P}$ if $u_1, \ldots, u_k \in P$, $v_1, \ldots, v_l \notin F$, and

$$P' = \{w \in P : X \notin sub(w)\} \cup (\mathbf{PER} \cap sub(uaYv))\,,$$
$$F' = \{w \in F : X \notin sub(w)\} \cup (\mathbf{FOR} \cap sub(uaYv))\,;$$

[6] A context-free production $A \to \alpha$ is linear if α contains no more than one occurrence of nonterminal symbols, i.e., $\alpha \in T^*NT^* \cup \{\varepsilon\}$.

[7] A context-free grammar with only linear productions.

2. For all productions $(X \to Yb, \{u_1, \ldots, u_k\}, \{v_1, \ldots, v_l\})$, add
$$[X,P,F,u,v] \to [Y,P',F',u,head(bv)]b$$
into $\bar{P}$ if $u_1, \ldots, u_k \in P$, $v_1, \ldots, v_l \notin F$, and
$$P' = \{w \in P : X \notin sub(w)\} \cup (\mathbf{PER} \cap sub(uYbv)),$$
$$F' = \{w \in F : X \notin sub(w)\} \cup (\mathbf{FOR} \cap sub(uYbv));$$
3. For all productions $(X \to Y, \{u_1, \ldots, u_k\}, \{v_1, \ldots, v_l\})$, add
$$[X,P,F,u,v] \to [Y,P',F',u,v]$$
into $\bar{P}$ if $u_1, \ldots, u_k \in P$, $v_1, \ldots, v_l \notin F$, and
$$P' = \{w \in P : X \notin sub(w)\} \cup (\mathbf{PER} \cap sub(uYv)),$$
$$F' = \{w \in F : X \notin sub(w)\} \cup (\mathbf{FOR} \cap sub(uYv));$$
4. For all productions $(X \to \varepsilon, \{u_1, \ldots, u_k\}, \{v_1, \ldots, v_l\})$, add
$$[X,P,F,u,v] \to \varepsilon$$
into $\bar{P}$ if $u_1, \ldots, u_k \in P$ and $v_1, \ldots, v_l \notin F$.

In the rest of this proof, we prove that $[X,P,F,u,v]$ says that the current sentential form in G contains X as its nonterminal, P is the set of all elements of permitting (**PER**) and F of all forbidding (**FOR**) contexts occurring in the current sentential form, and u and v are substrings of length no more than m; u is the left context of X and v is the right context.

Clearly, $[S,\emptyset,\emptyset,\varepsilon,\varepsilon]$ satisfies this condition. Let
$$w_1[X,P,F,u,v]w_2 \Rightarrow w_1a[Y,P',F',tail(ua),v]w_2$$
be a derivation step in $\bar{G}$, i.e. $u = tail(w_1)$ and $v = head(w_2)$. Let $P = \mathbf{PER} \cap sub(w_1Xw_2)$ and $F = \mathbf{FOR} \cap sub(w_1Xw_2)$. It is not hard to see that this derivation step is possible if and only if
$$w_1Xw_2 \Rightarrow w_1aYw_2$$
in G. We prove that
$$P' = \mathbf{PER} \cap sub(w_1aYw_2).$$
Clearly, $P' \subseteq \mathbf{PER} \cap sub(w_1aYw_2)$.

To prove the other inclusion, assume that $w \in \mathbf{PER} \cap sub(w_1aYw_2)$. Consider that it is possible that $X = Y$. If $w \in P$ and $X \notin sub(w)$, then $w \in P'$. If $w \in P$ and $X \in sub(w)$, i.e. $X = Y$, then $w \in sub(uaYv)$ and, therefore, $w \in P'$. Finally, assume that $w \notin P$. Then, $|w| \leq m$ and
$$w \in sub(tail(w_1)aYhead(w_2)) = sub(uaYv).$$

Thus, $w \in P'$ if and only if $w \in \mathbf{PER} \cap sub(w_1 aY w_2)$. Analogously for other types of productions, and also for the proof of $F' = \mathbf{FOR} \cap sub(w_1 aY w_2)$. These parts of the proof are left to the reader.

Because the previous holds for $[S, \emptyset, \emptyset, \varepsilon, \varepsilon] \Rightarrow \gamma$, i.e. $\emptyset = \mathbf{PER} \cap sub(S)$, $\emptyset = \mathbf{FOR} \cap sub(S)$, and S is a sentential form in G, the proof follows by induction to the length of the derivation. □

The following corollary is an immediate consequence of the previous theorem.

Corollary 5.2.4. *Context-conditional grammars having only linear productions are as powerful as linear grammars.*

Proof. In this case, only regular productions are used in the construction. □

5.2.2 Simple Context-Conditional Grammars

Consider a context-conditional grammar. If for each its production, either the permitting or the forbidding context is empty, then the grammar is called simple context-conditional. Formal definition follows.

Definition 5.2.5. Let $G = (N, T, P, S)$ be a context-conditional grammar. If

$$(X \rightarrow \alpha, Per, For) \in P$$

implies that

$$\emptyset \in \{Per, For\},$$

then G is said to be a *simple context-conditional grammar*.

We can easily prove the following theorem.

Theorem 5.2.6. *Every recursively enumerable language is generated by a simple context-conditional grammar of degree $(2,1)$ with no more than seven conditional productions and eight nonterminals.*

Proof. Let L be a recursively enumerable language. Then, there is a grammar

$$G_1 = (\{S, A, B, C\}, T, P \cup \{ABC \rightarrow \varepsilon\}, S)$$

in the first Geffert normal form such that $L = L(G_1)$. Construct the grammar

$$G = (\{S, A, B, C, A', B', C', B''\}, T, P' \cup P'', S),$$

where

$$P' = \{(X \rightarrow \alpha, \emptyset, \emptyset) : X \rightarrow \alpha \in P\},$$

and P'' contains the following seven conditional productions:

1. $(A \rightarrow A', \emptyset, \{A', B''\})$,
2. $(B \rightarrow B', \emptyset, \{B', B''\})$,
3. $(C \rightarrow C', \emptyset, \{C', B''\})$,

4. $(B' \to B'', \{A'B', B'C'\}, \emptyset)$,
5. $(A' \to \varepsilon, \{B''\}, \emptyset)$,
6. $(C' \to \varepsilon, \{B''\}, \emptyset)$,
7. $(B'' \to \varepsilon, \emptyset, \{A', C'\})$.

To prove that $L(G_1) \subseteq L(G)$, consider a derivation $S \Rightarrow^* wABCw'v \Rightarrow ww'v$ in G_1 by productions from P and the only one production $ABC \to \varepsilon$, where $w, w' \in \{A,B,C\}^*$ and $v \in T^*$. Then, $S \Rightarrow^* wABCw'v$ in G by productions from P'. By productions 1, 2, 3, 4, 5, 6, and 7,

$$\begin{aligned} wABCw'v &\Rightarrow wA'BCw'v \\ &\Rightarrow wA'B'Cw'v \\ &\Rightarrow wA'B'C'w'v \\ &\Rightarrow wA'B''C'w'v \\ &\Rightarrow wB''C'w'v \\ &\Rightarrow wB''w'v \\ &\Rightarrow ww'v. \end{aligned}$$

The inclusion follows by induction.

To prove that $L(G_1) \supseteq L(G)$, consider a terminal derivation. Notice that to eliminate a nonterminal, there must be B'' in the derivation. From production 4 and the observation that there is no more than one A', B', C' in the derivation (see productions 1, 2, 3), there cannot be a terminal between any two nonterminals. Therefore, the derivation is of the form $S \Rightarrow^* w_1w_2w_3$ in G by productions from P', where $w_1 \in \{A, AB\}^*$, $w_2 \in \{BC, C\}^*$, and $w_3 \in T^*$, and $w_1w_2w_3 \Rightarrow^* w_3$. Note that before S is eliminated, there is no occurrence of the substring ABC in the derivation. Then, $S \Rightarrow^* w_1w_2w_3$ in G_1 by productions from P. We prove that $w_1w_2w_3 \Rightarrow^* w_3$ in G_1.

For $w_1w_2 = \varepsilon$, the proof is done. For $w_1w_2 \neq \varepsilon$, there is B in w_1w_2; otherwise, B'' cannot be obtained and no nonterminal can be eliminated. To obtain B'', production 4 is applied. Therefore, $w_1w_2 = wABCw'$, where $w \in \{A, AB\}^*$ and $w' \in \{BC, C\}^*$; otherwise, the conditions of production 4 are not met. Thus, at the beginning, only productions 1, 2, and 3 are applicable. Then, only production 4 is applicable, and, after that, only productions 5 and 6 are applicable. Finally, only production 7 is applicable;

$$wABCw'w_3 \Rightarrow^3 wA'B'C'w'w_3 \Rightarrow wA'B''C'w'w_3 \Rightarrow^2 wB''w'w_3 \Rightarrow ww'w_3\,.$$

Thus, if $S \Rightarrow^* w_1w_2w_3 \Rightarrow^* w_3$ in G, where $w_1 \in \{A, AB\}^*$, $w_2 \in \{BC, C\}^*$, and $w_3 \in T^*$, then $S \Rightarrow^* w_1w_2w_3 \Rightarrow^* w_3$ in G_1. □

5.2.3 Generalized Forbidding Grammars

A generalized forbidding grammar is a context-conditional grammar, where the permitting context of any production is empty. These grammars are introduced in [Med90a]. (A few modifications of these grammars can be found in [CV92, CVM93, EKR94, Med90b].)

Definition 5.2.7. Let $G = (N, T, P, S)$ be a context-conditional grammar. If

$$(X \rightarrow \alpha, Per, For) \in P$$

implies that

$$Per = \emptyset\,,$$

then G is said to be a *generalized forbidding grammar*.

As all permitting contexts are empty, we simplify the notation as follows.

Notation 2. As far as generalized forbidding grammars are concerned, we omit the symbol $\emptyset$ from the notation of productions and, thus, write $(X \rightarrow \alpha, For)$ instead of $(X \rightarrow \alpha, \emptyset, For)$.

Notation 3. G is said to have *degree i* if G has degree (k,i) as a context-conditional grammar, for some k.

The last known result is the following theorem proved in [MŠ03].

Theorem 5.2.8. *Every recursively enumerable language is generated by a generalized forbidding grammar of degree two with no more than thirteen conditional productions and fifteen nonterminals.*

Now, we prove the main results of this section. First, however, we prove the following auxiliary lemma.

Lemma 5.2.9. *Let $L \in RE$, $L = L(G_1)$, G_1 is a grammar in the second Geffert normal form. Then, there is a grammar*

$$G = (\{S, 0, 1, \$\}, T, P \cup \{0\$0 \rightarrow \$, 1\$1 \rightarrow \$, \$ \rightarrow \varepsilon\}, S)\,,$$

with P containing only context-free productions of the form

$S \rightarrow h(u)Sa$ *if $S \rightarrow uSa$ in G_1,*
$S \rightarrow h(u)Sh(v)$ *if $S \rightarrow uSv$ in G_1,*
$S \rightarrow h(u)\$h(v)$ *if $S \rightarrow uv$ in G_1,*

where $h : \{A, B, C, D\}^* \rightarrow \{0, 1\}^*$ is a homomorphism defined as

$$h(A) = h(B) = 0 \text{ and } h(C) = h(D) = 1\,,$$

such that $L(G) = L(G_1)$.

Proof. Any terminal derivation in G_1 is, after the application of $S \rightarrow uv$, of the form

$$\{A, C\}^* \{B, D\}^* T^*\,.$$

From this, any terminal derivation in G is, after generating \$, of the form

$$h(\{A, C\}^*)\$h(\{B, D\}^*)T^*\,.$$

It is easy to see that if the production $AB \rightarrow \varepsilon$ or $CD \rightarrow \varepsilon$ is applied in G_1, then the production $0\$0 \rightarrow \$$ or $1\$1 \rightarrow \$$ is applied in G, respectively, and vice versa. Moreover, the last production applied in G in any terminal derivation is $\$ \rightarrow \varepsilon$. □

Theorem 5.2.10. *Every recursively enumerable language is generated by a generalized forbidding grammar of degree two and index six with no more than ten conditional productions and nine nonterminals.*

Proof. Let L be a recursively enumerable language. Then, there is a grammar

$$G = (\{S, 0, 1, \$\}, T, P \cup \{0\$0 \to \$, 1\$1 \to \$, \$ \to \varepsilon\}, S)$$

such that $L = L(G)$ and P contains productions of the form shown in Lemma 5.2.9. Construct the grammar

$$G' = (\{S', Z, S, 0, 1, 0', 1', \$, \#\}, T, P' \cup P'', S'),$$

where P' contains productions of the form

$(S' \to ZSZ, \emptyset)$,
$(S \to uSZaZ, \emptyset)$ if $S \to uSa \in P$,
$(S \to uSv, \emptyset)$ if $S \to uSv \in P$,
$(S \to u\$v, \emptyset)$ if $S \to uv \in P$,

and P'' contains following ten conditional productions:

1. $(0 \to 0', \{0', 1', \#\})$,
2. $(1 \to 1', \{0', 1', \#\})$,
3. $(0 \to 0'1', \{1', \#\})$,
4. $(1 \to 1'0', \{0', \#\})$,
5. $(\$ \to \#, \{0\$, 1\$, Z\$, \$0, \$1, \$Z\})$,
6. $(0' \to \varepsilon, \{\$, S\})$,
7. $(1' \to \varepsilon, \{\$, S\})$,
8. $(\# \to \$, \{0', 1'\})$,
9. $(Z \to \varepsilon, \{\$, \#, S\})$,
10. $(\$ \to \varepsilon, \{0, 1, 0', 1'\})$,

To prove that $L(G) \subseteq L(G')$, consider a derivation $S \Rightarrow^* w\$w^R v$ in G using only productions from P, where $w \in \{0,1\}^*$ and $v \in T^*$. This can be derived in G' by productions from P' as $S' \Rightarrow^* Zw\$w^R Zv'$, where $h(v') = v$ for a homomorphism $h : (T \cup \{Z\})^* \to T^*$ defined as $h(a) = a$, for $a \in T$, and $h(Z) = \varepsilon$. If $w = \varepsilon$, then

$$Z\$Zv' \Rightarrow ZZv' \Rightarrow^* v,$$

by productions 10 and 9. If $w = w'0$, then

$$\begin{aligned}
Zw'0\$0w'^R Zv' &\Rightarrow Zw'0'\$0w'^R Zv' \\
&\Rightarrow Zw'0'\$0'1'w'^R Zv' \\
&\Rightarrow Zw'0'\#0'1'w'^R Zv' \\
&\Rightarrow Zw'\#0'1'w'^R Zv' \\
&\Rightarrow Zw'\#1'w'^R Zv' \\
&\Rightarrow Zw'\#w'^R Zv' \\
&\Rightarrow Zw'\$w'^R Zv'
\end{aligned}$$

by productions 1, 3, 5, 6, 6, 7, and 8. The case of $w = w'1$ is analogous. The inclusion follows by induction.

To prove that $L(G) \supseteq L(G')$, consider a terminal derivation in G'

$$S' \Rightarrow^* Zw_1\$w_2Zw_3\,,$$

by productions from P', and

$$Zw_1\$w_2Zw_3 \Rightarrow^* w\,,$$

where $w_1, w_2 \in \{0,1\}^*$ and $w \in T^*$. We prove that $w_3 \in (T \cup \{Z\})^*$.

Assume that $Z0$ or $Z1$ is in $sub(Zw_3)$. Then, to eliminate this 0 or 1, production 6 or 7 must be applied. To apply production 6 or 7, production 5 must be applied before. Then, however, there is 0, 1, or Z next to \$; indeed, there cannot be more than two $0'$s or $1'$s in the derivation (there is no more than either $0'$ and $0'1'$, or $1'$ and $1'0'$). Thus, $w_3 \in (T \cup \{Z\})^*$ and $w = h(w_3)$. Then,

$$S \Rightarrow^* w_1\$w_2h(w_3)$$

in G by productions from P. We prove that

$$w_1\$w_2h(w_3) \Rightarrow^* h(w_3)\,.$$

Assume that $w_1 = w_2 = \varepsilon$. Then, the only applicable production in G' is production 10. After production 10, only production 9 is applicable. Thus, $Z\$Zw_3 \Rightarrow ZZw_3 \Rightarrow^* h(w_3)$.

Assume that $\varepsilon \in \{w_1, w_2\}$ and $w_1 \neq w_2$. Then,

$$Zw_1\$w_2Zw_3 \in \{Z\$w_2Zw_3, Zw_1\$Zw_3\}\,.$$

In both cases, neither 0 nor 1 can be eliminated (see production 5).

By induction on the length of w_1, we prove that $w_1 = w_2^R$. The basic step has already been proved. Assume that

$$Zw_1\$w_2Zw_3 = Zw_1'0\$xw_2'Zw_3\,,$$

where $x \in \{0,1\}$. Then, only productions 1, 2, 3, 4 can be applied. Notice that production 1 or 2 is applied before production 3 or 4; otherwise, if production 3 or 4 is applied, then neither production 1 nor 2 is applicable. Moreover, if production 1 is applied, then only production 3 is applicable, and, similarly, if production 2 is applied, then only production 4 is applicable. According to production 5, 0\$ is rewritten by production 1 or 3. Therefore, 0 is rewritten by production 1 and x is rewritten by production 3, or vice versa. Thus, $x = 0$ and

$$Zw_1'0\$0w_2'Zw_3 \Rightarrow^2 Zw_1'0'\$0'1'w_2'Zw_3 \quad \text{or} \quad Zw_1'0'1'\$0'w_2'Zw_3.$$

Then, only production 5 is applicable;

$$\Rightarrow \; Zw_1'0'\#0'1'w_2'Zw_3 \quad \text{or} \quad Zw_1'0'1'\#0'w_2'Zw_3$$

and only productions 6 and 7 are applicable;

$$\Rightarrow^3 Zw_1'\#w_2'Zw_3$$

and only production 8 is applicable;

$$\Rightarrow \; Zw_1'\$w_2'Zw_3.$$

The proof for $Zw_1\$w_2Zw_3 = Zw_1'1\$xw_2'Zw_3$, where $x \in \{0,1\}$, is analogous. By the induction hypothesis, $w_1 = w_2^R$.

Thus, if $S' \Rightarrow^* Zw_1\$w_1^RZw_3 \Rightarrow^* h(w_3)$ in G', where $w_1 \in \{0,1\}^*$ and $w_3 \in (T \cup \{Z\})^*$, then $S \Rightarrow^* w_1\$w_1^Rh(w_3) \Rightarrow^* h(w_3)$ in G. □

By a modification of the grammar from the proof of Theorem 5.2.10, the index can be decreased.

Theorem 5.2.11. *Every recursively enumerable language is generated by a generalized forbidding grammar of degree two and index four with no more than eleven conditional productions and ten nonterminals.*

Proof. Let L be a recursively enumerable language. There is a grammar

$$G = (\{S, 0, 1, \$\}, T, P \cup \{0\$0 \rightarrow \$, 1\$1 \rightarrow \$, \$ \rightarrow \varepsilon\}, S)$$

such that $L = L(G)$ and P contains productions of the form shown in Lemma 5.2.9. Construct the grammar

$$G' = (\{S', Z, S, 0, 1, 0', 1', \$, \#, @\}, T, P' \cup P'', S'),$$

where P' contains productions of the form

$(S' \rightarrow ZSZ, \emptyset)$,
$(S \rightarrow uSZaZ, \emptyset)$ if $S \rightarrow uSa \in P$,
$(S \rightarrow uSv, \emptyset)$ if $S \rightarrow uSv \in P$,
$(S \rightarrow u\$v, \emptyset)$ if $S \rightarrow uv \in P$,

and P'' contains following eleven conditional productions:

1. $(0 \rightarrow 0', \{0', 1', @\})$,
2. $(1 \rightarrow 1', \{0', 1', @\})$,
3. $(\$ \rightarrow \#, \{0\$, 1\$, Z\$\})$,
4. $(0 \rightarrow 0'1', \{0'1', 1', @\})$,
5. $(1 \rightarrow 1'0', \{1'0', 0', @\})$,
6. $(\# \rightarrow @, \{\#0, \#1, \#Z\})$,
7. $(0' \rightarrow \varepsilon, \{\$, \#, S\})$,
8. $(1' \rightarrow \varepsilon, \{\$, \#, S\})$,
9. $(@ \rightarrow \$, \{0', 1'\})$,
10. $(Z \rightarrow \varepsilon, \{\$, \#, @, S\})$,
11. $(\$ \rightarrow \varepsilon, \{0, 1\})$,

To prove that $L(G) \subseteq L(G')$, consider a derivation $S \Rightarrow^* w\$w^R v$ in G using only productions from P, where $w \in \{0,1\}^*$ and $v \in T^*$. This can be derived in G' by productions from P' as $S' \Rightarrow^* Zw\$w^R Zv'$, where $h(v') = v$ for a homomorphism $h : (T \cup \{Z\})^* \to T^*$ defined as $h(a) = a$, for $a \in T$, and $h(Z) = \varepsilon$. If $w = \varepsilon$, then

$$Z\$Zv' \Rightarrow ZZv' \Rightarrow^* v\,,$$

by productions 11 and 10. If $w = w'0$, then

$$\begin{aligned} Zw'0\$0w'^R Zv' &\Rightarrow Zw'0'\$0w'^R Zv' \\ &\Rightarrow Zw'0'\#0w'^R Zv' \\ &\Rightarrow Zw'0'\#0'1'w'^R Zv' \\ &\Rightarrow Zw'0'@0'1'w'^R Zv' \\ &\Rightarrow Zw'@0'1'w'^R Zv' \\ &\Rightarrow Zw'@1'w'^R Zv' \\ &\Rightarrow Zw'@w'^R Zv' \\ &\Rightarrow Zw'\$w'^R Zv' \end{aligned}$$

by productions 1, 3, 4, 6, 7, 7, 8, and 9. The case of $w = w'1$ is analogous. The inclusion follows by induction. Hence, if $S \Rightarrow^* v$ in G, $v \in T^*$, then $S' \Rightarrow^* v$ in G'.

To prove that $L(G) \supseteq L(G')$, consider a terminal derivation in G'

$$S' \Rightarrow^* Zw_1\$w_2Zw_3\,,$$

by productions from P', and

$$Zw_1\$w_2Zw_3 \Rightarrow^* w\,,$$

where $w_1, w_2 \in \{0,1\}^*$ and $w \in T^*$. We prove that $w_3 \in (T \cup \{Z\})^*$.

Assume that $Z0$ or $Z1$ is in $sub(Zw_3)$. Then, to eliminate this 0 or 1, production 7 or 8 is applied to this 0 or 1. To apply production 7 or 8, production 3 or 6 is applied before. However, there is 0, 1, or Z next to \$ or #; indeed, there cannot be more than two $0'$s or $1'$s in the derivation—a contradiction; production 3 or 6 cannot be applied. Thus, $w_3 \in (T \cup \{Z\})^*$ and $w = h(w_3)$. Then,

$$S \Rightarrow^* w_1\$w_2h(w_3)$$

in G by productions from P. We prove that

$$w_1\$w_2h(w_3) \Rightarrow^* h(w_3)\,.$$

Assume that $w_1 = w_2 = \varepsilon$. Then, the only applicable production is production 11 followed by production 10. Clearly, $\$h(w_3) \Rightarrow h(w_3)$ in G.

Assume that $\varepsilon \in \{w_1, w_2\}$ and $w_1 \neq w_2$. Then,

$$Zw_1\$w_2Zw_3 \in \{Z\$w_2Zw_3, Zw_1\$Zw_3\}\,.$$

In both cases, neither 0 nor 1 can be eliminated.

By induction on the length of w_1, we prove that $w_1 = w_2^R$. The basic step has already been proved. Assume that

$$Zw_1\$w_2Zw_3 = Zw_1'0\$xw_2'Zw_3\,,$$

where $x \in \{0,1\}$. Then, only productions 1, 2, 4, 5 are applicable. Notice that production 1 (2) has to be applied before 4 (5); otherwise, if production 4 (5) is applied, then production 1 (2) is not applicable. Moreover, if production 1 is applied, then only production 4 is applicable, and if production 2 is applied, then only production 5 is applicable. According to production 3, 0\$ is rewritten by production 1 or 4. Therefore, 0 is rewritten by production 1 and x is rewritten by production 4, or vice versa. Thus, $x = 0$ and

$$\begin{aligned} Zw_1'0\$0w_2'Zw_3 \Rightarrow^2 Zw_1'0'\$0'1'w_2'Zw_3 \quad &\text{or} \quad Zw_1'0'1'\$0'w_2'Zw_3 \\ &\text{or} \quad Zw_1'0'\#0w_2'Zw_3 \text{ (by 1 and 3).} \end{aligned}$$

Then, only production 3 or 4 is applicable;

$$\Rightarrow \; Zw_1'0'\#0'1'w_2'Zw_3 \quad \text{or} \quad Zw_1'0'1'\#0'w_2'Zw_3$$

and only production 6 is applicable;

$$\Rightarrow \; Zw_1'0'@0'1'w_2'Zw_3 \text{ or } \quad Zw_1'0'1'@0'w_2'Zw_3$$

and only productions 7 and 8 are applicable;

$$\Rightarrow^3 Zw_1'@w_2'Zw_3$$

and only production 9 is applicable;

$$\Rightarrow^3 Zw_1'\$w_2'Zw_3.$$

The proof for $Zw_1\$w_2Zw_3 = Zw_1'1\$xw_2'Zw_3$, where $x \in \{0,1\}$, is analogous. By the induction hypothesis, $w_1 = w_2^R$.

Thus, if $S' \Rightarrow^* Zw_1\$w_1^RZw_3 \Rightarrow^* h(w_3)$ in G', where $w_1 \in \{0,1\}^*$ and $w_3 \in (T \cup \{Z\})^*$, then $S \Rightarrow^* w_1\$w_1^Rh(w_3) \Rightarrow^* h(w_3)$ in G. □

In the following two theorems, we decrease the number of nonterminals and the number of conditional productions disregarding the index.

Theorem 5.2.12. *Every recursively enumerable language is generated by a generalized forbidding grammar of degree two and index nine with no more than eight conditional productions and ten nonterminals.*

Proof. Let L be a recursively enumerable language. Then, there is a grammar

$$G_1 = (\{S,A,B,C\},T,P \cup \{ABC \to \varepsilon\},S)$$

in the first Geffert normal form such that $L = L(G_1)$. Construct the grammar

$$G = (\{S,S',Z,A,B,C,A',B',C',\#\},T,P' \cup P'',S)\,,$$

where P' contains productions of the form

$(S \to ZS'Z, \emptyset)$,
$(S' \to uS'ZaZ, \emptyset)$ if $S \to uSa \in P$,
$(S' \to uS'v, \emptyset)$ if $S \to uSv \in P$,
$(S' \to uv, \emptyset)$ if $S \to uv \in P$,

and P'' contains the following eight conditional productions:

1. $(A \to \#A', \{\#, S'\})$,
2. $(B \to B', \{B', \#, S'\})$,
3. $(C \to C', \{C', \#, S'\})$,
4. $(A' \to \varepsilon, \{A'\}\{A, B, C, C', Z\})$,
5. $(B' \to \varepsilon, \{B'\}\{A, B, C, Z\} \cup \{A, B, C, C', Z\}\{B'\})$,
6. $(C' \to \varepsilon, \{A', B'\} \cup \{A, B, C, Z\}\{C'\})$,
7. $(\# \to \varepsilon, \{A', B', C'\})$,
8. $(Z \to \varepsilon, \{S', A, A', B, B', C, C'\})$.

To prove that $L(G_1) \subseteq L(G)$, consider a derivation $S \Rightarrow^* wABCw'v \Rightarrow ww'v$ in G_1 by productions from P and the only one application of the production $ABC \to \varepsilon$, where $w, w' \in \{A, B, C\}^*$ and $v \in T^*$. Then, $S \Rightarrow^* ZwABCw'Zv'$ in G by productions from P', where $v' \in (T \cup \{Z\})^*$ is such that $h(v') = v$, for a homomorphism $h : (T \cup \{Z\})^* \to T^*$ defined as $h(a) = a$, for $a \in T$, and $h(Z) = \varepsilon$. By productions 3, 2, 1, 4, 5, 6, and 7,

$$\begin{aligned} ZwABCw'Zv' &\Rightarrow ZwABC'w'Zv' \\ &\Rightarrow ZwAB'C'w'Zv' \\ &\Rightarrow Zw\#A'B'C'w'Zv' \\ &\Rightarrow Zw\#B'C'w'Zv' \\ &\Rightarrow Zw\#C'w'Zv' \\ &\Rightarrow Zw\#w'Zv' \\ &\Rightarrow Zww'Zv'. \end{aligned}$$

The inclusion follows by induction and, eventually, by production 8.

To prove that $L(G_1) \supseteq L(G)$, observe that if there is a string of the form $Z\{B', C'\}$ as a substring of a sentential form, then neither of productions 5 and 6 is applicable to the rightmost nonterminal of this string—there is Z before the nonterminal. Thus, we can assume that

$$S \Rightarrow^* Zw_1w_2Zw_3$$

in G, by productions from P', and that

$$Zw_1w_2Zw_3 \Rightarrow^* h(w_3)\,,$$

where $w_1 \in \{A, AB\}^*$, $w_2 \in \{BC, C\}^*$, and $w_3 \in (T \cup \{Z\})^*$. Notice that before S and S' are eliminated, there is no occurrence of ABC in the sentential form (see [Gef88a]), and, moreover, no production from P'' can be applied. Then, $S \Rightarrow^* w_1w_2h(w_3)$ in G_1 by productions from P. We prove that

$$w_1w_2h(w_3) \Rightarrow^* h(w_3).$$

By induction on the length of w_1w_2, we prove that $w_1w_2 = w'_1ABCw'_2$, for some $w'_1 \in \{A, AB\}^*$ and $w'_2 \in \{BC, C\}^*$, or $w_1w_2 = \varepsilon$. In any derivation step, there is no more than one A', B', C', and no X', for $X \in \{A, B, C\}$, is generated while there is # in the sentential form (see productions 1, 2, 3). Moreover, # is eliminated after all primed nonterminals are eliminated (see production 7). We prove that A, B, and C are in $sub(w_1w_2)$, for $w_1w_2 \neq \varepsilon$.

1. $A \in sub(w_1w_2)$: to eliminate A, A has to be rewritten to A'. Then, B' has to follow A' (by production 4) and C' has to follow B' (by production 5).
2. $B \in sub(w_1w_2)$: to eliminate B, B has to be rewritten to B'. Then, A' or # has to be before B' and C' has to follow B' (by production 5).
3. $C \in sub(w_1w_2)$: to eliminate C, C has to be rewritten to C'. Then, # has to be before C' (by production 6)—that is, $A \in sub(w_1w_2)$; otherwise, this case is analogical to 1.

In all above cases, we have $ABC \in sub(w_1w_2)$. Thus, $w_1w_2 = w'_1ABCw'_2$, for some $w'_1 \in \{A, AB\}^*$ and $w'_2 \in \{BC, C\}^*$.

We prove that while ABC is eliminated, no other nonterminal is eliminated, and then # is removed.

First, only productions 1, 2, and 3 are applicable.

(i) If production 1 is applied, then productions 2 and 3 are not applicable because there is # in the sentential form. Also, production 4 is not applicable because A' is followed by A, B, C, or Z. Thus, the derivation is blocked.

(ii) Assume that production 2 is applied first. Then, there is B' in the sentential form. Notice that production 5 is not applicable because B' is followed by A, B, C, or Z. Thus, only productions 1 and 3 are applicable. To apply production 5, # or A' has to be before B' and C' has to follow B'. If production 1 is applied, then production 3 is not applicable—C' cannot be generated. Moreover, if there is $\#A'B'\{A, B, C, Z\}$ as a substring of the sentential form, then A' can be eliminated (by production 4). However, no other production is applicable. Thus, the sequence of productions in the derivation is 2, 3, and 1.

(iii) Assume that production 3 is applied first. Then, there is C' in the sentential form. Notice that production 6 is not applicable because A, B, C, or Z is before C'. To apply production 6, # has to be before C'. Thus, only productions 1 and 2 are applicable. If production 1 is applied, then production 2 is not applicable. To eliminate A', A' has to be followed by B' (see production 4)—a contradiction; there is no B' in the sentential form. Therefore, production 2 had to be applied before production 1. Thus, the sequence of productions in the derivation is 3, 2, and 1.

After the sequence of productions 2, 3, 1, or 3, 2, 1, productions 4 and 5 are applicable if and only if $\#A'B'C'$ is a substring of the sentential form (see productions 4 and 5). Notice that no other productions are applicable. Thus,

$$w'_1ABCw'_2h(w_3) \Rightarrow^2 w'_1AB'C'w'_2h(w_3) \Rightarrow w'_1\#A'B'C'w'_2h(w_3).$$

After the application of productions 4 and 5 (in this order, otherwise A' cannot be eliminated),

$$w_1'\#A'B'C'w_2'h(w_3) \Rightarrow w_1'\#B'C'w_2'h(w_3) \Rightarrow w_1'\#C'w_2'h(w_3)\,,$$

only production 6 is applicable,

$$w_1'\#C'w_2'h(w_3) \Rightarrow w_1'\#w_2'h(w_3)\,.$$

If $w_1'w_2' \neq \varepsilon$, then only production 7 is applicable because there is no A', B', C' in the sentential form. If $w_1'w_2' = \varepsilon$, then also production 8 is applicable. However, it is easy to see that it does not matter whether some Zs are eliminated before # is removed. Then,

$$w_1'\#w_2'h(w_3) \Rightarrow^+ w_1'w_2'h(w_3)\,.$$

As a result, by the induction hypothesis,

$$w_1'ABCw_2'h(w_3) \Rightarrow^* w_1'w_2'h(w_3) \Rightarrow^* h(w_3)\,.$$

Thus, if $S \Rightarrow^* Zw_1w_2Zw_3 \Rightarrow^* h(w_3)$ in G, where $w_1 \in \{A,AB\}^*$, $w_2 \in \{BC,C\}^*$, and $w_3 \in (T \cup \{Z\})^*$, then $S \Rightarrow^* w_1w_2h(w_3) \Rightarrow^* h(w_3)$ in G_1. Hence, the other inclusion holds. □

If we allow the index to have no limit, then the number of nonterminals can be decreased. To prove this, we first need to modify Lemma 5.2.9. More precisely, only the homomorphism h is modified.

Lemma 5.2.13. *Let $L \in RE$, $L = L(G_1)$, G_1 is a grammar in the second Geffert normal form. Then, there is a grammar*

$$G = (\{S,0,1,\$\},T,P \cup \{0\$0 \to \$, 1\$1 \to \$, \$ \to \varepsilon\},S)$$

with P containing only context-free productions of the form

$S \to h(u)Sa$ *if $S \to uSa$ in G_1,*
$S \to h(u)Sh(v)$ *if $S \to uSv$ in G_1,*
$S \to h(u)\$h(v)$ *if $S \to uv$ in G_1,*

where $h : \{A,B,C,D\}^* \to \{0,1\}^*$ is a homomorphism defined as

$$h(A) = h(B) = 00,\ h(C) = 01,\ \textit{and}\ h(D) = 10\,,$$

such that $L(G) = L(G_1)$.

Now, we can prove the following theorem.

Theorem 5.2.14. *Every recursively enumerable language is generated by a generalized forbidding grammar of degree two and unlimited index with no more than nine conditional productions and eight nonterminals.*

Proof. Let L be a recursively enumerable language. Then, there is a grammar

$$G = (\{S,0,1,\$\},T,P \cup \{0\$0 \to \$, 1\$1 \to \$, \$ \to \varepsilon\},S)$$

such that $L = L(G)$ and P contains productions of the form shown in Lemma 5.2.13. Construct the grammar

$$G' = (\{S', S, 0, 1, 0', 1', \$, \#\}, T, P' \cup P'', S'),$$

where P' contains productions of the form

$(S' \to 111S11, \emptyset)$,
$(S \to uS11a, \emptyset)$ if $S \to uSa \in P$,
$(S \to uSv, \emptyset)$ if $S \to uSv \in P$,
$(S \to u\$v, \emptyset)$ if $S \to uv \in P$,

and P'' contains following nine conditional productions:

1. $(0 \to 0', \{0', 1', \#\})$,
2. $(1 \to 1', \{0', 1', \#\})$,
3. $(0 \to 0'1', \{1', \#\})$,
4. $(1 \to 1'0', \{0', \#\})$,
5. $(\$ \to \#, \{0\$, 1\$, \$0, \$1\} \cup \{\$\}T)$,
6. $(0' \to \varepsilon, \{\$, S\})$,
7. $(1' \to \varepsilon, \{\$, S\})$,
8. $(\# \to \$, \{0', 1'\})$,
9. $(\$ \to \varepsilon, \{0, 0'\})$,

To prove that $L(G) \subseteq L(G')$, consider a derivation $S \Rightarrow^* w\$w^R v$ in G using only productions from P, where $w \in \{00, 01\}^*$ and $v \in T^*$. This can be derived in G' by productions from P' as $S' \Rightarrow^* 111w\$w^R 11v'$, where $v' \in (T\{11\})^*$ and $h(v') = v$ for a homomorphism $h : (T \cup \{1\})^* \to T^*$ defined as $h(a) = a$, for $a \in T$, and $h(1) = \varepsilon$. If $w = \varepsilon$, then

$$111\$11v' \Rightarrow 11111v' \Rightarrow^* v,$$

by productions 9, and repeating productions 2 and 7. If $w = w'0$, then

$$\begin{aligned}
111w'0\$0w'^R 11v' &\Rightarrow 111w'0'\$0w'^R 11v' \\
&\Rightarrow 111w'0'\$0'1'w'^R 11v' \\
&\Rightarrow 111w'0'\#0'1'w'^R 11v' \\
&\Rightarrow 111w'\#0'1'w'^R 11v' \\
&\Rightarrow 111w'\#1'w'^R 11v' \\
&\Rightarrow 111w'\#w'^R 11v' \\
&\Rightarrow 111w'\$w'^R 11v'
\end{aligned}$$

by productions 1, 3, 5, 6, 6, 7, and 8. The case of $w = w'1$ is analogous. The inclusion follows by induction.

To prove that $L(G) \supseteq L(G')$, consider a terminal derivation in G'

$$S' \Rightarrow^* 111w_1\$w_2 11w_3$$

by productions from P', and

$$111w_1\$w_2 11w_3 \Rightarrow^* w\,,$$

where $w_1 \in \{00,01\}^*$, $w_2 \in \{00,10\}^*$, and $w \in T^*$.

Assume that $\varepsilon \in \{w_1, w_2\}$ and $w_1 \neq w_2$. Then,

$$111w_1\$w_2 11w_3 \in \{111\$w_2 11w_3, 111w_1\$11w_3\}\,.$$

First, assume that

$$111\$w_2 11w_3 = 111\$xw_2' 11w_3\,,$$

where $x \in \{00,10\}$. As in the proof of Theorem 5.2.10, only productions 1, 2, 3, and 4 can be applied. Moreover, production 1 (or 2) is applied before production 3 (or 4), and if production 1 is applied, then only production 3 is applicable, and, similarly, if production 2 is applied, then only production 4 is applicable. According to production 5, 1\$ is rewritten by production 2 or 4. Therefore, 1 is rewritten by production 2 and x is rewritten by production 4, or vice versa. Thus, $x = 10$ and

$$111\$10w_2' 11w_3 \Rightarrow^7 11\$0w_2' 11w_3\,.$$

Similarly, assume that $111w_1\$11w_3 = 111w_1'x\$11w_3$, $x \in \{00,01\}$. Then, $x = 01$ and

$$111w_1'01\$11w_3 \Rightarrow^* 111w_1'0\$1w_3\,.$$

In both cases, the derivation is blocked.

Assume that $w_1 = w_2 = \varepsilon$, i.e. $S' \Rightarrow^* 111\$11w_3$, where $w_3 = aw_3'$, for some $a \in T$, or $w_3 = \varepsilon$. Then,

$$111\$11w_3 \Rightarrow^* \alpha\,,$$

where

$$\alpha \in \{111\$11w_3, 11\$1w_3, 1\$aw_3', 1\$\}\,.$$

In all cases, to remove \$, production 9 is applied. However, production 9 is applicable if and only if there is no 0 in w_3. Clearly, $\$w \Rightarrow w$ in G.

Analogously to the proof of Theorem 5.2.10, by induction on the length of w_1, we can prove that $w_1 = w_2^R$.

Thus, we have proved that $0 \notin sub(w_3)$, i.e. $w = h(w_3)$, and, moreover, if

$$S' \Rightarrow^* 111w_1\$w_1^R 11w_3 \Rightarrow^* h(w_3)$$

in G', where $w_1 \in \{00,01\}^*$, then

$$S \Rightarrow^* w_1\$w_1^R h(w_3) \Rightarrow^* h(w_3)$$

in G. $\square$

5.2.4 Generalized Permitting Grammars

A generalized permitting grammar is a context-conditional grammar, where the forbidding context of any production is empty.

Definition 5.2.15. Let $G = (N, T, P, S)$ be a context-conditional grammar. If

$$(X \to \alpha, Per, For) \in P$$

implies that

$$For = \emptyset\,,$$

then G is said to be a *generalized permitting grammar*.

As all forbidding contexts are empty, we simplify the notation as follows.

Notation 4. As far as generalized permitting grammars are concerned, we omit the symbol $\emptyset$ from the notation of productions and, thus, write $(X \to \alpha, Per)$ instead of $(X \to \alpha, Per, \emptyset)$.

Notation 5. G is said to have *degree i* if G has degree (i, k) as a context-conditional grammar, for some k.

Open Problem

The question whether generalized permitting grammars characterize the whole family of recursively enumerable languages or not is a long-standing open problem.

5.2.5 Semi-Conditional Grammars

A semi-conditional grammar is a context-conditional grammar, where both permitting and forbidding contexts contain no more than one element. These grammars are introduced and studied in [Pău85].

Definition 5.2.16. A *semi-conditional grammar*, G, is a quadruple

$$G = (N, T, P, S)\,,$$

where

- N is a nonterminal alphabet,
- T is a terminal alphabet such that $N \cap T = \emptyset$,
- $S \in N$ is the start symbol, and
- P is a finite set of productions of the form

  $$(X \to \alpha, u, v)$$

  with $X \in N$, $\alpha \in (N \cup T)^*$, and $u, v \in (N \cup T)^+ \cup \{0\}$, where $0 \notin N \cup T$ is a special symbol. If $u \neq 0$ or $v \neq 0$, then the production $(X \to \alpha, u, v) \in P$ is said to be *conditional*.

G has *degree* (i,j) if for all productions $(X \to \alpha, u, v) \in P$,

$$u \neq 0 \text{ implies } |u| \leq i$$

and

$$v \neq 0 \text{ implies } |v| \leq j\,.$$

For $x_1, x_2 \in (N \cup T)^*$, $x_1 X x_2$ *directly derives* $x_1 \alpha x_2$ according to the production $(X \to \alpha, u, v) \in P$, denoted by $x_1 X x_2 \Rightarrow x_1 \alpha x_2$, if

$$u \neq 0 \text{ implies that } u \in sub(x)$$

and

$$v \neq 0 \text{ implies that } v \notin sub(x)\,.$$

As usual, $\Rightarrow$ is extended to $\Rightarrow^i$, for $i \geq 0$, $\Rightarrow^+$, and $\Rightarrow^*$. The language generated by a semi-conditional grammar, G, is defined as

$$L(G) = \{w \in T^* : S \Rightarrow^* w\}\,.$$

We now prove the main result concerning descriptional complexity of semi-conditional grammars.

Theorem 5.2.17. *Every recursively enumerable language is generated by a semi-conditional grammar of degree* $(2,1)$ *with no more than seven conditional productions and eight nonterminals.*

Proof. Let L be a recursively enumerable language. There is a grammar

$$G = (\{S, A, B, C\}, T, P \cup \{ABC \to \varepsilon\}, S)$$

in the first Geffert normal form such that $L = \mathscr{L}(G)$. Construct the grammar

$$G' = (\{S, A, B, C, A', B', C', \$\}, T, P' \cup P'', S)\,,$$

where

$$P' = \{(X \to \alpha, 0, 0) : X \to \alpha \in P\}\,,$$

and P'' contains following seven conditional productions:

1. $(A \to \$A', 0, \$)$,
2. $(B \to B', A', B')$,
3. $(C \to C'\$, A'B', C')$,
4. $(B' \to \varepsilon, B'C', 0)$,
5. $(C' \to \varepsilon, A'C', 0)$,
6. $(A' \to \varepsilon, A'\$, 0)$,
7. $(\$ \to \varepsilon, 0, A')$.

To prove that $\mathscr{L}(G) \subseteq \mathscr{L}(G')$, consider a derivation

$$S \Rightarrow^* wABCw'v \Rightarrow ww'v$$

in G by productions from P with only one application of the production $ABC \to \varepsilon$, where $w, w' \in \{A, B, C\}^*$ and $v \in T^*$. Then,

$$S \Rightarrow^* wABCw'v$$

in G' by productions from P'. Moreover, by productions 1, 2, 3, 4, 5, 6, 7, 7, we get

$$\begin{aligned}
wABCw'v &\Rightarrow w\$A'BCw'v \\
&\Rightarrow w\$A'B'Cw'v \\
&\Rightarrow w\$A'B'C'\$w'v \\
&\Rightarrow w\$A'C'\$w'v \\
&\Rightarrow w\$A'\$w'v \\
&\Rightarrow w\$\$w'v \\
&\Rightarrow w\$w'v \\
&\Rightarrow ww'v.
\end{aligned}$$

The inclusion follows by induction.

To prove that $\mathscr{L}(G) \supseteq \mathscr{L}(G')$, consider a terminal derivation. Let $X \in \{A, B, C\}$ be in a sentential form of this derivation. To eliminate X, there are following three possibilities:

1. If $X = A$, then there must be C and B (by productions 6 and 3) in the derivation;
2. If $X = B$, then there must be C and A (by productions 4 and 3) in the derivation;
3. If $X = C$, then there must be A and B (by productions 5 and 3) in the derivation.

In all above cases, there are A, B, and C in the derivation. By productions 1, 2, 3, and 7, there cannot be more than one A', B', and C' in any sentential form of this terminal derivation. Moreover, by productions 3 and 4, $A'B'C'$ is a substring of a sentential form of this terminal derivation, and there is no terminal symbol between any two nonterminals; otherwise, there will be a situation in which (at least) one of productions 3 and 4 will not be applicable. Thus, any first part of a terminal derivation in G' is of the form

$$S \Rightarrow^* w_1ABCw_2w \Rightarrow^3 w_1\$A'B'C'\$w_2w \tag{5.1}$$

by productions from P' and productions 1, 2, and 3, where $w_1 \in \{A, B\}^*$, $w_2 \in \{B, C\}^*$, and $w \in T^*$. Next, only production 4 is applicable. Thus,

$$\Rightarrow w_1\$A'C'\$w_2w\,.$$

Besides a possible application of production 2, only production 5 is applicable. Thus,

$$\Rightarrow^+ w_1'\$A'\$w_2'w,$$

where $w_1' \in \{A,B,B'\}^*$, $w_2' \in \{B,B',C\}^*$. Besides a possible application of production 2, only production 6 is applicable. Thus,

$$\Rightarrow^+ w_1''\$\$w_2''w,$$

where $w_1'' \in \{A,B,B'\}^*$, $w_2'' \in \{B,B',C\}^*$. Finally, only production 7 is applicable, i.e.,

$$\Rightarrow^2 w_1''w_2''w\,.$$

Thus, by productions 1, 2, 3, or 1, 3, if production 2 has already been applied, we get

$$\Rightarrow^* uvw\,.$$

Here,

$$uvw \in \{u_1\$A'B'C'\$u_2w : u_1 \in \{A,B\}^*, u_2 \in \{B,C\}^*\}$$

or $uv = \varepsilon$.

Thus, the substring ABC and only this substring was eliminated during the previous derivation. By induction (see (5.1)), the inclusion holds. This derivation can be performed in G with an application of the production $ABC \to \varepsilon$, too. □

Note that it is well-known that every recursively enumerable language is generated by a semi-conditional grammar of degree $(1,1)$ (see Theorems 6 and 11(b) in [May72]). In this case, however, no limit of the number of nonterminals or conditional productions is known.

5.2.6 Simple Semi-Conditional Grammars

A simple semi-conditional grammar is a semi-conditional grammar, where for each production, either the permitting or the forbidding context does not contain any element. These grammars are introduced in [GM94].

Definition 5.2.18. Let $G = (N,T,P,S)$ be a semi-conditional grammar. If

$$(X \to \alpha, u, v) \in P$$

implies that

$$0 \in \{u, v\}\,,$$

then G is said to be a *simple semi-conditional grammar*.

The last known result concerning descriptional complexity of simple semi-conditional grammars is the content of the following theorem proved in [Vas05].

Theorem 5.2.19. *Every recursively enumerable language is generated by a simple semi-conditional grammar of degree* $(2,1)$ *with no more than ten conditional productions and twelve nonterminals.*

We improve this result as follows.

Theorem 5.2.20. *Every recursively enumerable language is generated by a simple semi-conditional grammar of degree* $(2,1)$ *with no more than nine conditional productions and ten nonterminals.*

Proof. Let L be a recursively enumerable language. Then, there is a grammar

$$G_1 = (\{S,A,B,C\},T,P \cup \{ABC \to \varepsilon\},S)$$

in the first Geffert normal form such that $L = L(G_1)$. Construct the grammar

$$G = (\{S,A,B,C,A',B',C',\$,B'',C''\},T,P' \cup P'',S),$$

where

$$P' = \{(S \to \alpha,0,0) : S \to \alpha \in P\}$$

and P'' contains the following nine conditional productions:

1. $(A \to A',0,A')$,
2. $(B \to B',0,B')$,
3. $(C \to C',0,C')$,
4. $(B' \to B'',A'B',0)$,
5. $(C' \to C'',B''C',0)$,
6. $(B'' \to \varepsilon,B''C'',0)$,
7. $(A' \to \$,A'C'',0)$,
8. $(C'' \to \varepsilon,\$,0)$,
9. $(\$ \to \varepsilon,0,C'')$.

To prove that $L(G_1) \subseteq L(G)$, consider a derivation $S \Rightarrow^* wABCw'v \Rightarrow ww'v$ in G_1 by productions from P with only one application of the production $ABC \to \varepsilon$, where $w,w' \in \{A,B,C\}^*$ and $v \in T^*$. Then, $S \Rightarrow^* wABCw'v$ in G by productions from P. By productions 3, 2, 1, 4, 5, 6, 7, 8, and 9,

$$\begin{aligned} wABCw'v &\Rightarrow wABC'w'v \\ &\Rightarrow wAB'C'w'v \\ &\Rightarrow wA'B'C'w'v \\ &\Rightarrow wA'B''C'w'v \\ &\Rightarrow wA'B''C''w'v \\ &\Rightarrow wA'C''w'v \\ &\Rightarrow w\$C''w'v \\ &\Rightarrow w\$w'v \\ &\Rightarrow ww'v. \end{aligned}$$

The inclusion follows by induction.

To prove that $L(G_1) \supseteq L(G)$, consider a terminal derivation. Let $X \in \{A,B,C\}$ be in a sentential form. To eliminate X, there are following three possibilities:

1. if $X = A$, then there has to be C (by production 7) and B (by production 5) in the sentential form;
2. if $X = B$, then there has to be A (by production 4) and C (by production 6) in the sentential form;
3. if $X = C$, then there has to be B (by production 5) and A (by production 8) in the sentential form.

In all above cases, there are A, B, and C in the sentential form. By productions 1, 2, and 3, there can be no more than one A', B', and C' in the sentential form. By productions 4 and 5, A' is before B' and C' follows this B'. We prove that in any terminal derivation, there is no terminal symbol between any two nonterminals. More precisely, there is no substring of the form $T\{BC,C\}$. Assume that aB, for some $a \in T$, is a substring of the sentential form. Then, B is rewritten to B' and B' cannot be rewritten to B'' because A' is before aB'. Similarly, if there is aC in the sentential form, for some $a \in T$, then C is rewritten to C' and aC' cannot be rewritten to aC'' because there is never B'' followed by C'. Thus, any terminal derivation in G is of the form

$$S \Rightarrow^* w_1A'w_2B'w_3C'w_4w \tag{5.2}$$

by productions from P and productions 1, 2, 3, and

$$\Rightarrow^* w,$$

where $w_1 \in \{A,B\}^*$, $w_2, w_3 \in \{A,B,C,S\}^*$, $w_4 \in \{B,C\}^*$, and $w \in T^*$. We prove that $S \notin sub(w_2w_3)$. To rewrite B' (by production 4), $w_2 = \varepsilon$. Thus,

$$w_1A'B'w_3C'w_4w \Rightarrow w_1A'B''w_3C'w_4w \tag{5.3}$$

and, also, production 2 is applicable. However, to rewrite C' (by production 5), $w_3 = \varepsilon$. Thus,

$$\Rightarrow^+ w_1A'B''C''w_4w,$$

where $w_1 \in \{A,B,B'\}^*$, $w_4 \in \{B,B',C\}^*$. Thus, we have that $A'B'C'$ is a substring of $w_1A'w_2B'w_3C'w_4w$, and $A'B'C'$ was obtained from ABC.

Next, we prove that no other nonterminal is eliminated while ABC is eliminated. Besides a possible application of productions 2 and 3, only production 6 is applicable. Thus,

$$\Rightarrow^+ w_1A'C''w_4w,$$

where $w_1 \in \{A,B,B'\}^*$, $w_4 \in \{B,B',C,C'\}^*$. Besides a possible application of productions 2 and 3, only production 7 is applicable. Thus,

$$\Rightarrow^+ w_1\$C''w_4w,$$

where $w_1 \in \{A,B,B'\}^*$, $w_4 \in \{B,B',C,C'\}^*$. Besides a possible application of productions 1, 2, 3, and 4, only production 8 is applicable. Thus,

$$\Rightarrow^{+} w_1\$w_4w,$$

where $w_1 \in \{A, A', A'B'', B, B'\}^*$, $w_4 \in \{B, B', C, C'\}^*$. Besides a possible application of productions 1, 2, 3, and 4, only production 9 is applicable. Thus,

$$\Rightarrow^{+} w_1w_4w,$$

where $w_1 \in \{A, A', A'B'', B, B'\}^*$, $w_4 \in \{B, B', C, C'\}^*$. Thus,

$$\Rightarrow^{*} uvw$$

by productions 1, 2, and 3, if they are applicable. Then,

$$uvw \in \{u_1A'B'C'u_4w : u_1 \in \{A, B\}^*, u_4 \in \{B, C\}^*\}$$
$$\cup\{v_1A'B''C'v_4w : v_1 \in \{A, B, B'\}^*, v_4 \in \{B, B', C\}^*\}$$

or $uv = \varepsilon$.

Thus, the string ABC, and only the string, was eliminated. By induction (see (5.2) and (5.3)), the inclusion holds. This derivation can be performed in G_1 with an application of the production $ABC \to \varepsilon$, too. □

In [MŠ05], the question what is the generative power of simple semi-conditional grammars of degree $(1,1)$ is formulated as an open problem. Recently, we have proved that simple semi-conditional grammars of degree $(1,1)$ characterize the whole family of recursively enumerable languages (see [MMb]).

Theorem 5.2.21. *Simple semi-conditional grammars of degree* $(1,1)$ *characterize the family of all recursively enumerable languages.*

6

Conclusion

This monograph discusses the study of formal languages and automata. Its main contribution consists in the following results proved in this monograph and published (or submitted) in [Mas06, Mas07b, MM07a, MM07c, MMa, MMb, MM07b, MM07d].

(I) The first part of this monograph, Chapter 4, introduces and studies two variants of self-regulating finite automata, which have a close relation to some parallel grammars, and which with respect to the number of turns made during their computations define an infinite proper hierarchy of language families in the family of context-sensitive languages.

In the conclusion of the chapter, self-regulating pushdown automata are mentioned and studied. A proof that the hierarchy of language families accepted by n-turn all-move self-regulating pushdown automata, for $n \in \mathbb{N}_0$, collapses on $n = 1$ is given. In that case, it is shown that one-turn all-move self-regulating pushdown automata possess the power of Turing machines, whereas it is easy to see that zero-turn all-move (and, in the same way, first-move) self-regulating pushdown automata possess exactly the power of pushdown automata. However, as far as first-move self-regulating pushdown automata are concerned, the question whether the hierarchy of language families accepted by n-turn first-move self-regulating pushdown automata, for $n \in \mathbb{N}_0$, collapses as well or not and what is the power of k-turn first-move self-regulating pushdown automata, for some $k \in \mathbb{N}$, is an open problem.

More specifically, based on the number of turns, Chapter 4 of this monograph proves that

1. n-turn first-move self-regulating finite automata give rise to an infinite proper hierarchy of language families coinciding with the hierarchy resulting from $(n+1)$-parallel right linear grammars;
2. n-turn all-move self-regulating finite automata give rise to an infinite proper hierarchy of language families coinciding with the hierarchy resulting from $(n+1)$-right linear simple matrix grammars;
3. all-move self-regulating pushdown automata do not give rise to any infinite hierarchy analogical to hierarchies resulting from the self-regulating finite automata.

Moreover, it is shown that while zero-turn all-move self-regulating pushdown automata define the family of context-free languages, one-turn all-move self-regulating pushdown automata define the family of recursively enumerable languages.

Although this monograph has solved the main problems concerning self-regulating finite and pushdown automata, there still remain some problems open. Perhaps the most important open problems are included in 1 through 3 given next.

1. What is the language family accepted by n-turn first-move self-regulating pushdown automata, when $n \in \mathbb{N}$?
2. By analogy with standard deterministic finite and pushdown automata, introduce the deterministic versions of self-regulating finite and pushdown automata. What is their power?
3. Discuss the closure properties under other language operations, such as the reversal.

(II) The second part of this monograph, Chapter 5, studies descriptional complexity of partially parallel grammars and grammars regulated by context conditions, which are regulated context-free grammars. Results concerning descriptional complexity of these grammars are supplemented and improved in this monograph. It is shown that very limited number of nonterminals and special (conditional) productions is needed.

First, recall the known results that every recursively enumerable language is generated

(1) by a scattered context grammar with no more than five nonterminals and two non-context-free productions;
(2) by a multisequential grammar with no more than six nonterminals;
(3) by a multicontinuous grammar with no more than six nonterminals;
(4) by a context-conditional grammar (without any limit to the number of conditional productions and nonterminals);
(5) by a simple context-conditional grammar (without any limit to the number of conditional productions and nonterminals);
(6) by a generalized forbidding grammar of degree two with no more than thirteen conditional productions and fifteen nonterminals;
(7) by a semi-conditional grammar (without any limit to the number of conditional productions and nonterminals); and
(8) by a simple semi-conditional grammar of degree $(2,1)$ with no more than ten conditional productions and twelve nonterminals.

This monograph improves the previous results and proves that every recursively enumerable language is generated

(A) by a scattered context grammar with no more than four non-context-free productions and four nonterminals;
(B) by a multisequential grammar with no more than two selectors and two nonterminals;

(C) by a multicontinuous grammar with no more than two selectors and three nonterminals;
(D) by a context-conditional grammar of degree $(2,1)$ with no more than six conditional productions and seven nonterminals;
(E) by a simple context-conditional grammar of degree $(2,1)$ with no more than seven conditional productions and eight nonterminals;
(F) by a generalized forbidding grammar of degree two and index six with no more than ten conditional productions and nine nonterminals;
(G) by a generalized forbidding grammar of degree two and index four with no more than eleven conditional productions and ten nonterminals;
(H) by a generalized forbidding grammar of degree two and index nine with no more than eight conditional productions and ten nonterminals;
(I) by a generalized forbidding grammar of degree two and unlimited index with no more than nine conditional productions and eight nonterminals;
(J) by a semi-conditional grammar of degree $(2,1)$ with no more than seven conditional productions and eight nonterminals; and
(K) by a simple semi-conditional grammar of degree $(2,1)$ with no more than nine conditional productions and ten nonterminals.

However, the question whether these results achieved in this monograph can be established for fewer nonterminals or conditionals productions with the same (or even less) degree is open.

Index

Acceptance, 11
Accepted language, 11
Alphabet, 5

Chomsky hierarchy, 9
Closed under operation, 7
Complement, 6
Component, 23
Computation, 11
Concatenation, 5, 6
Configuration, 11, 12

Degree, 57
Descriptional complexity, 41
Direct derivation, 8

Equivalence of languages, 8
Extended Post correspondence problem, 43

Finite automaton, 11

Generated language, 8
Grammar, 8
  Context-conditional, 57
  Context-free, 9
  Context-sensitive, 9
  Forbidding, 59
  Generalized forbidding, 64
  Generalized permitting, 75
  Multicontinuous, 55
  Multisequential, 53
  Parallel right linear, 23
  Permitting, 59
  Random context, 59
    with appearance checking, 59
  Regular, 9
  Right linear simple matrix, 28
  Scattered context, 43
    Propagating, 47
  Semi-conditional, 75
  Simple context-conditional, 62
  Simple semi-conditional, 78
  Type 0, 9

Homomorphism, 7
  Inverse, 7

Index, 57

Kleene closure, 7

Language, 6
  Finite, 6
  Infinite, 6
  Parallel right linear, 23
  Right linear simple matrix, 29
Linear language, 14

Normal form
  First Geffert, 9
  Kuroda, 9
  Penttonen, 9
  Second Geffert, 10
  Third Geffert, 10

Positive closure, 7
Power, 6
Power of a string, 5

Prefix, 6
Production
  Conditional, 57
  Context-free, 43
  Context-sensitive, 43
Pushdown automaton, 12

Reversal, 5, 6
Right quotient, 7
Rule labels, 11

Selector, 53
Self-regulating automaton, 19
  All-move self-regulating finite automaton, 21
  All-move self-regulating pushdown automaton, 22
  First-move self-regulating finite automaton, 20
  First-move self-regulating pushdown automaton, 22
Sentence, 8
Sentential form, 8
String, 5
  Empty, 5
  Substring, 6
Substitution, 7
  Finite, 7
  Nonerasing, 7
Suffix, 6
Symbol, 5

Terminal derivation, 8
Turn state, 19

References

[ABB97] J. M. Autebert, J. Berstel, and L. Boasson. Context-free languages and pushdown automata. In G. Rozenberg and A. Salomaa, editors, *Handbook of Formal Languages*, volume 1, Word Language Grammar, pages 111–174. Springer-Verlag, Berlin, 1997.

[AU72] A. V. Aho and J. D. Ullman. *The Theory of Parsing, Translation and Compiling, Volume I: Parsing*. Prentice Hall, Englewood Cliffs, New Jersey, 1972.

[Bak72] B. S. Baker. Context-sesitive grammars generating context-free languages. In M. Nivat, editor, *Automata, Languages and Programming*, pages 501–506. North-Holland, Amsterdam, 1972.

[BF96] H. Bordihn and H. Fernau. Accepting grammars and systems: an overview. In J. Dassow, G. Rozenberg, and A. Salomaa, editors, *Developments in Language Theory II; at the crossroads of mathematics, computer science and biology*, pages 199–208. Singapore: World Scientific, 1996.

[Boo72] R. V. Book. Terminal context in context-sensitive grammars. *SIAM Journal on Computing*, 1:20–30, 1972.

[CJK88] M. Chytil, L. Janiga, and V. Koubek, editors. *Mathematical Foundations of Computer Science 1988, MFCS'88, Carlsbad, Czechoslovakia, August 29 – September 2, 1988, Proceedings*, volume 324 of *Lecture Notes in Computer Science*. Springer, 1988.

[Cou77] B. Courcelle. On jump deterministic pushdown automata. *Mathematical Systems Theory*, 11:87–109, 1977.

[CV92] E. Csuhaj-Varju. On grammars with local and global context conditions. *International Journal of Computer Mathematics*, 47:17–27, 1992.

[CVM93] E. Csuhaj-Varju and A. Meduna. Grammars with context conditions. *EATCS Bulletin*, 53:199–213, 1993.

[DFP99] J. Dassow, H. Fernau, and Gh. Păun. On the leftmost derivation in matrix grammars. *International Journal of Foundations of Computer Science*, 10(1):61–80, 1999.

[DP89] J. Dassow and Gh. Păun. *Regulated Rewriting in Formal Language Theory*. Springer-Verlag, Berlin, 1989.

[EKR94] A. Ehrenfeucht, J. Kleijn, and G. Rozenberg. Adding global forbidding context to context-free grammars. *Theoretical Computer Science*, 31:161–206, 1994.

[ER79] A. Ehrenfeucht and G. Rozenberg. An observation on scattered grammars. *Information Processing Letters*, 9(2):84–85, 1979.

[Fer96] H. Fernau. Scattered context grammars with regulation. *Ann. Univ. Bucharest, Math.-Informatics Series*, 45:41–49, 1996.

[Fer00] H. Fernau. Regulated grammars under leftmost derivation. *Grammars*, 3(1):37–62, 2000.

[FP94] M. Frazier and C. D. Page. Prefix grammars: An alternative characterization of the regular languages. *Information Processing Letters*, 51(2):67–71, 1994.

[FR68] P. C. Fischer and A. L. Rosenberg. Multitape one-way nonwriting automata. *Journal of Computer and System Sciences*, 2:38–101, 1968.

[Gef88a] V. Geffert. Context-free-like forms for the phrase-structure grammars. In Chytil et al. [CJK88], pages 309–317.

[Gef88b] V. Geffert. A representation of recursively enumerable languages by two homomorphisms and a quotient. *Theoretical Computer Science*, 62:235–249, 1988.

[Gef91a] V. Geffert. How to generate languages using only two pairs of parentheses. *Journal of Information Processes in Cybernetics EIK*, 27:303–315, 1991.

[Gef91b] V. Geffert. Normal forms for phrase-structure grammars. *Theoretical Informatics and Applications*, 25(5):473–496, 1991.

[GG66] S. Ginsburg and S. Greibach. Mappings which preserve context-sensitive languages. *Information and Control*, 9:563–582, 1966.

[GGH67] S. Ginsburg, S. A. Greibach, and M. A. Harrison. One-way stack automata. *Journal of the ACM*, 14:389–418, 1967.

[GH69] S. Greibach and J. Hopcroft. Scattered context grammars. *Journal of Computer and System Sciences*, 3:233–247, 1969.

[Gin75] S. Ginsburg. *Algebraic and Automata-theoretic Properties of Formal Languages*. North-Holland, Amsterdam, 1975.

[GM94] A. Gopalaratnam and A. Meduna. On semi-conditional grammars with productions having either forbidding or permitting conditions. *Acta Cybernetica*, 11(4):307–324, 1994.

[GP07] V. Geffert and G. Pighizzini, editors. *Proceedings of the 9th International Workshop on Descriptional Complexity of Formal Systems, DCFS'07*. High Tatras, Slovakia, 2007.

[Gre69] S. A. Greibach. Checking automata and one-way stack languages. *Journal of Computer and System Sciences*, 3:196–217, 1969.

[GS68] S. Ginsburg and E. Spanier. Finite-turn pushdown automata. *SIAM Journal on Control and Optimization*, 4:429–453, 1968.

[GW89] J. Gonczarowski and M. K. Warmuth. Scattered versus context-sensitive rewriting. *Acta Informatica*, 27(1):81–95, 1989.

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

[Hib74] T. N. Hibbard. Context-limited grammars. *Journal of the ACM*, 21:446–453, 1974.

[HU79] J. E. Hopcroft and J. O. Ullman. *Introduction to Automata Theory, Languages and Computation*. Addison-Wesley, 1979.

[Iba70] O. H. Ibarra. Simple matrix languages. *Information and Control*, 17:359–394, 1970.

[Ito04] M. Ito. *Algebraic Theory of Automata and Languages*. World Scientific, Singapore, 2004.

[Kel84] J. Kelemen. Conditional grammars: Motivations, definitions, and some properties. In *Proc. Conf. Automata, Languages and Mathematical Sciences*, pages 110–123, Salgótarján, 1984.

[KR83a] H. C. M. Kleijn and G. Rozenberg. Multi grammars. *International Journal of Computer Mathematics*, 12:177–201, 1983.

[KR83b] H. C. M. Kleijn and G. Rozenberg. On the generative power of regular pattern grammars. *Acta Informatica*, 20:391–411, 1983.

[LP81] H. R. Lewis and C. H. Papadimitriou. *Elements of the Theory of Computation*. Prentice-Hall, Englewood Cliffs, 1981.

[LRCP06] H. Leung, D. Ranjan, B. Cloteaux, and G. Pighizzini, editors. *Proceedings of the 8th International Workshop on Descriptional Complexity of Formal Systems, DCFS'06*. Las Cruces, New Mexico, USA, 2006.

[Mar91] J. C. Martin. *Introduction to Languages and the Theory of Computation*. McGraw-Hill, New York, 1991.

[Mas06] T. Masopust. An improvement of the descriptional complexity of grammars regulated by context conditions. In *Second Doctoral Workshop on Mathematical and Engineering Methods in Computer Science (MEMICS 2006)*, pages 105–112, Mikulov, 2006.

[Mas07a] T. Masopust. Generalized forbidding grammars with linear productions. In *Third Doctoral Workshop on Mathematical and Engineering Methods in Computer Science (MEMICS 2007)*, pages 121–126, Znojmo, CZ, 2007.

[Mas07b] T. Masopust. A note on the descriptional complexity of semi-conditional grammars. In *Proceedings of the 2nd workshop on formal models (WFM 2007)*, pages 213–218, Hradec nad Moravicí, 2007.

[Mas07c] T. Masopust. Scattered context grammars can generate the powers of 2. In *Proceedings of the 13th Conference and Competition STUDENT EEICT 2007 Volume 4*, pages 401–404, Brno, 2007.

[May72] O. Mayer. Some restrictive devices for context-free grammars. *Information and Control*, 20:69–92, 1972.

[Med90a] A. Meduna. Generalized forbidding grammars. *International Journal of Computer Mathematics*, 36:31–38, 1990.

[Med90b] A. Meduna. Global context conditional grammars. *Journal of Automata, Languageas and Combinatorics*, pages 31–38, 1990.

[Med91] A. Meduna. Global context conditional grammars. *Journal of Information Processes in Cybernetics EIK*, 27:159–165, 1991.

[Med94] A. Meduna. Matrix grammars under leftmost and rightmost restrictions. In Gh. Păun, editor, *Mathematical Linguistics and Related Topics*, pages 243–257. Romanian Academy of Sciences, Bucharest, 1994.

[Med95] A. Meduna. Syntactic complexity of scattered context grammars. *Acta Informatica*, 32:285–298, 1995.

[Med97a] A. Meduna. Four-nonterminal scattered context grammars characterize the family of recursively enumerable languages. *International Journal of Computer Mathematics*, 63:67–83, 1997.

[Med97b] A. Meduna. On the number of nonterminals in matrix grammars with leftmost derivations. In *New Trends in Formal Languages: Control, Cooperation, and Combinatorics*, pages 27–39. Springer-Verlag, 1997.

[Med97c] A. Meduna. Six-nonterminal multi-sequential grammars characterize the family of recursively enumerable languages. *International Journal of Computer Mathematics*, 65:179–189, 1997.

[Med98a] A. Meduna. Descriptional complexity of multi-continuous grammars. *Acta Cybernetica*, 13:375–384, 1998.

[Med98b] A. Meduna. Economical transformations of phrase-structure grammars to scattered context grammars. *Acta Cybernetica*, 13:375–384, 1998.

[Med00a] A. Meduna. *Automata and Languages: Theory and Applications*. Springer-Verlag, London, 2000.

[Med00b] A. Meduna. Generative power of three-nonterminal scattered context grammars. *Theoretical Computer Science*, 246:279–284, 2000.

[Med00c] A. Meduna. Terminating left-hand sides of scattered context productions. *Theoretical Computer Science*, 237:423–427, 2000.

[Med02] A. Meduna. Descriptional complexity of scattered rewriting and multirewriting: An overview. *Journal of Automata, Languageas and Combinatorics*, 7:571–577, 2002.

[Med03] A. Meduna. Simultaneously one-turn two-pushdown automata. *International Journal of Computer Mathematics*, 80:679–687, 2003.

[Med06] A. Meduna. Deep pushdown automata. *Acta Informatica*, 42(8–9):541–552, 2006.

[MF03a] A. Meduna and H. Fernau. On the degree of scattered context-sensitivity. *Theoretical Computer Science*, 290:2121–2124, 2003.

[MF03b] A. Meduna and H. Fernau. A simultaneous reduction of several measures of descriptional complexity in scattered context grammars. *Information Processing Letters*, 86:235–240, 2003.

[MK00] A. Meduna and D. Kolář. Regulated pushdown automata. *Acta Cybernetica*, 14:653–664, 2000.

[MK02] A. Meduna and D. Kolář. One-turn regulated pushdown automata and their reduction. *Fundamenta Informaticae*, 51:399–405, 2002.

[MMa] T. Masopust and A. Meduna. On the descriptional complexity of partially parallel grammars. Submitted.

[MMb] T. Masopust and A. Meduna. On the generative power of simple semi-conditional grammars. Submitted.

[MM07a] T. Masopust and A. Meduna. Descriptional complexity of generalized forbidding grammars. In *Proceedings of the Workshop on Descriptional Complexity of Formal Systems (DCFS 2007)*, pages 170–177, High Tatras, Slovakia, 2007.

[MM07b] T. Masopust and A. Meduna. Descriptional complexity of grammars regulated by context conditions. In *Pre-proceedings of the 1st International Conference on Language and Automata Theory and Applications (LATA 2007)*, pages 403–411, Tarragona, Spain, 2007.

[MM07c] T. Masopust and A. Meduna. Descriptional complexity of semi-conditional grammars. *Information Processing Letters*, 104:29–31, 2007.

[MM07d] A. Meduna and T. Masopust. Self-regulating finite automata. *Acta Cybernetica*, 18:135–153, 2007.

[MPPW05] C. Mereghetti, B. Palano, G. Pighizzini, and D. Wotschke, editors. *Proceedings of the 7th International Workshop on Descriptional Complexity of Formal Systems, DCFS'05*. Como, Italy, 2005.

[MR71] D. Milgram and A. Rosenfeld. A note on scattered context grammars. *Information Processing Letters*, 1(2):47–50, 1971.

[MS97] A. Mateescu and A. Salomaa. Aspects of classical language theory. In G. Rozenberg and A. Salomaa, editors, *Handbook of formal languages*, volume 1, Word Language Grammar, pages 175–251. Springer-Verlag, Berlin, 1997.

[MŠ02] A. Meduna and M. Švec. Reduction of simple semi-conditional grammars with respect to the number of conditional productions. *Acta Cybernetica*, 15:353–360, 2002.

[MŠ03] A. Meduna and M. Švec. Descriptional complexity of generalized forbidding grammars. *International Journal of Computer Mathematics*, 80:11–17, 2003.

[MŠ05] A. Meduna and M. Švec. *Grammars with Context Conditions and Their Applications*. John Wiley & Sons, New York, 2005.

[Pău82] Gh. Păun. On simple matrix languages versus scattered context languages. *RAIRO Operations Research*, 16:245–253, 1982.

[Pău85] Gh. Păun. A variant of random context grammars: Semi-conditional grammars. *Theoretical Computer Science*, 41:1–17, 1985.

[Pen74] M. Penttonen. One-sided and two-sided context in formal grammars. *Information and Control*, 25:371–392, 1974.

[Rev83] G. E. Revesz. *Introduction to Formal Language Theory*. McGraw-Hill, New York, 1983.

[Ros72] R. D. Rosebrugh. Restricted parallelism and regular grammars. Master's thesis, McMaster University, 1972.

[RS97] G. Rozenberg and A. Salomaa, editors. *Handbook of Formal Languages*, volume 2. Springer-Verlag, Berlin, 1997.

[RW73] R. D. Rosebrugh and D. Wood. A characterization theorem for n-parallel right linear languages. *Journal of Computer and System Sciences*, 7:579–582, 1973.

[RW75] R. D. Rosebrugh and D. Wood. Restricted parallelism and right linear grammars. *Utilitas Mathematica*, 7:151–186, 1975.

[Sak81] J. Sakarovitch. Pushdown automata with terminating languages. *Languages and Automata Symposium, RIMS 421, Kyoto University*, pages 15–29, 1981.

[Sal73] A. Salomaa. *Formal languages*. Academic Press, New York, 1973.

[Sir69] R. Siromoney. *Studies in the Mathematical Theory of Grammars and its Applications*. PhD thesis, University of Madras, Madras, India, 1969.

[Sir71] R. Siromoney. Finite-turn checking automata. *Journal of Computer and System Sciences*, 5:549–559, 1971.

[Sud88] T. A. Sudkamp. *Languages and Machines*. Addison-Wesley, Reading, Massachusetts, 1988.

[Val89] L. Valiant. The equivalence problem for deterministic finite turn pushdown automata. *Information and Control*, 81:265–279, 1989.

[Vas05] Gy. Vaszil. On the descriptional complexity of some rewriting mechanisms regulated by context conditions. *Theoretical Computer Science*, 330:361–373, 2005.

[vdW70] A. P. J. van der Walt. Random context grammars. In *Proceedings of the Symposium on Formal Languages*. 1970.

[Vir73] V. Virkkunen. On scattered context grammars. *Acta Univ. Oulu. Ser. A Sci. Rerum Natur.*, Mathematica 6:75–82, 1973.

[Woo73] D. Wood. Properties of n-parallel finite state languages. Technical report, McMaster University, 1973.

[Woo75] D. Wood. m-parallel n-right linear simple matrix languages. *Utilitas Mathematica*, 8:3–28, 1975.

Název	Formal Models: Regulation and Reduction
Autor	Mgr. Tomáš Masopust, Ph.D.

Vydavatel	Vysoké učení technické v Brně
	Fakulta informačních technologií
Obálka	Mgr. Dagmar Hejduková
Tisk	MJ servis, spol. s r.o.
Vyšlo	Brno, 2007
Vydání	první

Tato publikace neprošla redakční ani jazykovou úpravou.