BRNO UNIVERSITY OF TECHNOLOGY

Faculty of Information Technology

Department of Information Systems

Ing. Martin Švec

Grammars with Context Conditions and Their Applications

Gramatiky s kontextovými podmínkami a jejich aplikace

Short Version of Ph.D. Thesis

Study Field:	Information Systems
Supervisor:	Doc. RNDr. Alexander Meduna, CSc.
Opponents:	Prof. Ing. Tomáš Hruška, CSc. Prof. RNDr. Miroslav Novotný, CSc.
Presentation Date:	March 25, 2005

Keywords

formal language theory, regulated rewriting, generative power, descriptional complexity, grammars with context conditions, L grammars

Klíčová slova

teorie formálních jazyků, řízené přepisování, generativní síla, popisná složitost, gramatiky s kontextovými podmínkami, L systémy

The thesis is available in the library of the Faculty of Information Technology, Brno University of Technology, Božetěchova 2, 612 66 Brno.

ISBN 80–214–2905–4
ISSN 1213–4198

Contents

1 Introduction 5

2 Preliminaries and Definitions 7

3 Conditions Placed on Derivation Domains 9

4 Conditions Placed on the Use of Productions 11

4.1 Sequential Conditional Grammars . . . 11

4.1.1 Context-Conditional Grammars . . . 11

4.1.2 Random-Context Grammars . . . 12

4.1.3 Generalized Forbidding Grammars . . . 13

4.1.4 Simple Semi-Conditional Grammars . . . 13

4.2 Parallel Conditional Grammars . . . 14

4.2.1 Forbidding ET0L Grammars . . . 15

4.2.2 Simple Semi-Conditional ET0L Grammars . . . 16

5 Conditions Placed on the Neighborhood of Rewritten Symbols 17

5.1 Continuous Context . . . 17

5.2 Scattered Context . . . 17

6 Grammatical Transformations and Simulations 18

6.1 Derivation Simulation . . . 18

6.2 Grammatical Simulation . . . 20

6.3 Close Simulation of E(0,1)L Grammars by Symbiotic E0L Grammars 21

7 Applications and Implementation 21

8 Conclusion 26

References 28

Author's Curriculum Vitae 30

Abstract

The present thesis studies grammars with context conditions and their applications. In particular, it discusses sequential and parallel grammars whose derivation steps are restricted by some conditions placed on the rewritten sentential forms. According to the types of context conditions, it classifies the grammars with context conditions into three classes and sums up crucial results about them. Specifically, this classification results from the distinction between context conditions placed on (1) the domains of grammatical derivations, (2) the use of grammatical productions, and (3) the neighborhood of the rewritten symbols. In all these cases, the main attention is concentrated on establishing the grammatical generative power and important properties. In particular, this thesis studies how to reduce these grammars with respect to some of their components, such as the number of grammatical symbols or productions, in order to make the grammars small, succinct and, therefore, easy to use in practice. To demonstrate this practical use, it also discusses the applications and implementation of grammars with context conditions. Most of the applications are related to microbiology, which definitely belongs to the central application areas of computer science today.

1 Introduction

Formal languages fulfill a crucial role in many computer science areas, ranging from compilers through mathematical linguistics to molecular genetics. Whenever dealing with them, we face the problem of choosing their appropriate models in order to grasp them elegantly and precisely. By analogy with the specification of natural languages, we often base these models upon suitable grammars.

A grammar generates its language by performing derivation steps that change strings, called sentential forms, to other strings according to its grammatical productions. During a derivation step, the grammar rewrites a part of its current sentential form with a string according to one of its productions. If in this way it can make a sequence of derivation steps from its start symbol to a sentential form consisting of terminal symbols—that is, the symbols over which the language is defined, the resulting sentential form is called a sentence and belongs to the generated language. The set of all sentences made in this way is the language generated by the grammar.

In the classical formal language theory, we can divide grammatical productions into context-dependent and context-independent productions, and based on this division, we can naturally distinct context-dependent grammars, such as phrase-structure grammars, from context-independent grammars, such as context-free grammars. Making a derivation step according to context-dependent productions depends on rather strict conditions, usually placed on the context surrounding the rewritten symbol while making a step according to context-independent productions does not. From this point of view, we obviously tend to use context-independent grammars. Unfortunately, compared to context-dependent grammars, context-independent grammars are significantly less powerful; in fact, most of them are incapable to grasp some aspects of quite common programming languages. On the other hand, most context-dependent grammars are as powerful as the Turing machines, and this remarkable power represents their indisputable advantage. To overcome the difficulties and, at the same time, maintain the advantages described above, the modern language theory has introduced some new grammars that simultaneously satisfy these three properties:

- they are based on context-independent productions;
- their context conditions are signficantly more simple and flexible than the strict condition placed on the context surrounding the rewritten symbol in the classical context-dependent grammars;
- they are as powerful as classical context-dependent grammars.

In this thesis, we overview the most essential types of these grammars, whose alternative context conditions can be classified into these three categories:

- context conditions placed on derivation domains;

- context conditions placed on the use of productions;
- context conditions placed on the neighborhood of the rewritten symbols.

As already pointed out, we want the context conditions as small as possible. Therefore, we pay a lot of attention to the reduction of context conditions in this study. Specifically, we reduce the number of some of their components, such as the number of nonterminals or productions. We study how to achieve this reduction without any decrease of their generative power. By achieving this reduction, we actually make the grammars with context conditions more succinct and economical, and these properties are obviously highly appreciated both from a practical and theoretical standpoint. Regarding each of the dicussed grammars, we introduce and study their parallel and sequential versions, which represent two basic approaches to grammatical generation of languages in today's formal language theory. To be more specific, during a sequential derivation step, a grammar rewrites a single symbol in the current sentential form while during a parallel derivation step, a grammar rewrites all symbols. As context-free and E0L grammars represent perhaps the most fundamental sequential and parallel grammars, respectively, we usually base the discussion of sequential and parallel generation of languages on them.

Organization

The thesis consists of the following sections:

Section 2 gives an introduction to formal languages and their grammars.

Section 3 restricts grammatical derivation domains in a very simple and natural way. Under these restrictions, both sequential and parallel context-independent grammars characterize the family of recursively enumerable languages, which are defined by the Turing machines.

Section 4 studies grammars with conditional use of productions. In these grammars, productions may be applied on condition that some symbols occur in the current sentential form and some others do not. We discuss many sequential and parallel versions of these grammars in detail. Most importantly, new characterizations of some well-known families of L languages, such as the family of ET0L languages, are obtained.

Section 5 studies grammars with context conditions placed on the neighborhood of rewritten symbols. We distinguish between scattered and continuous context neighborhood. The latter strictly requires that the neighborhood of the rewritten symbols forms a continuous part of the sentential form while the former drops this requirement of continuity.

Section 6 takes a closer look at grammatical transformations, which are frequently studied in the previous chapters. Specifically, it studies how to transform grammars with context-conditions to some other equivalent grammars so that both the input grammars and the transformed grammars generate their languages in a very similar way.

Section 7 demostrates the use of grammars with context conditions by several applications related to biology.

Section 8 summarizes the main results of the thesis and states several open problems. In addition, it proposes new directions in the investigation of these grammars.

2 Preliminaries and Definitions

This section reviews the basics of grammars. Specifically, it provides definitions of context-free, context-sensitive, and phrase-structure grammars along with some related notions and basic results which are used throughout the thesis.

Definition 1. A *phrase-structure* grammar is a quadruple $G = (V, T, P, S)$, where V is the *total alphabet*, T is the set of *terminals* ($T \subset V$), $P \subseteq V^*(V - T)V^* \times V^*$ is a finite relation, and $S \in V - T$ is the *axiom* of G. The symbols in $V - T$ are referred to as *nonterminals*. In what follows, each $(x, y) \in P$ is called a *production* or a *rule* and written as $x \to y \in P$; accordingly, P is called the *set of productions* in G. The relation of a *direct derivation* in G is a binary relation over V^* denoted by $\Rightarrow_G$ and defined in the following way. Let $x \to y \in P$, $u, v, z_1, z_2 \in V^*$, and $u = z_1xz_2$, $v = z_1yz_2$; then, $u \Rightarrow_G v\ [x \to y]$. When no confusion exists, we simplify $u \Rightarrow_G v\ [x \to y]$ to $u \Rightarrow_G v$. By $\Rightarrow_G^k$, we denote the k-fold product of $\Rightarrow_G$. Furthermore, let $\Rightarrow_G^+$ and $\Rightarrow_G^*$ denote the transitive closure of $\Rightarrow_G$ and the transitive and reflexive closure of $\Rightarrow_G$, respectively. If $S \Rightarrow_G^* x$ for some $x \in V^*$, x is called a *sentential form*. If $S \Rightarrow_G^* w$, where $w \in T^*$, $S \Rightarrow_G^* w$ is said to be a *successful derivation* of G. The *language of* G, denoted by $L(G)$, is defined as $L(G) = \{w \in T^* : S \Rightarrow_G^* w\}$. In the literature, the phrase-structure grammars are also often defined with productions of the form $xAy \to xuy$, where $u, x, y \in V^*$, $A \in V - T$ (see [3]). Both definitions are interchangeable in the sense that the grammars defined in these two ways generate the same family of languages—the family of *recursively enumerable languages*, denoted by **RE**.

Definition 2. A *context-sensitive grammar* is a phrase-structure grammar, $G = (V, T, P, S)$, such that each production in P is of the form $xAy \to xuy$, where $A \in V - T$, $u \in V^+$, $x, y \in V^*$. A *context-sensitive language* is a language generated by a context-sensitive grammar. The family of context-sensitive languages is denoted by **CS**.

Definition 3. A *context-free grammar* is a phrase-structure grammar, $G = (V, T, P, S)$, such that each production $x \to y \in P$ satisfies $x \in V - T$. A *context-free language* is a language generated by a context-free grammar. The family of context-free languages is denoted by **CF**.

For the families of languages generated by context-free, context-sensitive and phrase-structure grammars, it holds:

Theorem 1 (see [9]). $\mathbf{CF} \subset \mathbf{CS} \subset \mathbf{RE}$.

Besides context-free, context-sensitive and phrase-structure grammars, we also discuss ET0L grammars and EIL grammars in this study.

Definition 4. An *ET0L grammar* (see [21], [22]) is a t+3-tuple, $G = (V, T, P_1, \ldots, P_t, S)$, where $t \geq 1$, and V, T, and S are the total alphabet, the terminal alphabet ($T \subset V$), and the axiom ($S \in V - T$), respectively. Each P_i is a finite set of productions of the form $a \to x$, where $a \in V$ and $x \in V^*$. If $a \to x \in P_i$ implies $x \neq \varepsilon$ for all $i \in \{1, \ldots, t\}$, G is said to be *propagating* (an *EPT0L grammar* for short). Let $u, v \in V^*$, $u = a_1a_2 \ldots a_q$, $v = v_1v_2 \ldots v_q$, $q = |u|$, $a_j \in V$, $v_j \in V^*$, and $p_1, p_2, \ldots, p_q$ is a sequence of productions of the form $p_j = a_j \to v_j \in P_i$ for all $j = 1, \ldots, q$, for some $i \in \{1, \ldots, t\}$. Then, u *directly derives* v according to the productions p_1 through p_q, denoted by $u \Rightarrow_G v\ [p_1, p_2, \ldots, p_q]$. In the standard manner, we define the relations $\Rightarrow_G^k$ ($k \geq 0$), $\Rightarrow_G^+$, and $\Rightarrow_G^*$. The language of G, denoted by $L(G)$, is defined as $L(G) = \{w \in T^* : S \Rightarrow_G^* w\}$. The families of languages generated by ET0L and EPT0L grammars are denoted by **ET0L** and **EPT0L**, respectively.

Let $G = (V, T, P_1, \ldots, P_t, S)$ be an ET0L grammar. If $t = 1$, G is called an *E0L grammar*. We denote the families of languages generated by E0L and propagating E0L grammars (EP0L grammars for short) by **E0L** and **EP0L**, respectively.

By **E0L**, **EP0L**, **ET0L**, and **EPT0L**, we denote the families of languages generated by E0L grammars, EP0L grammars, and EPT0L grammars, respectively.

Theorem 2 (see [21]). $\mathbf{CF} \subset \mathbf{E0L} = \mathbf{EP0L} \subset \mathbf{ET0L} = \mathbf{EPT0L} \subset \mathbf{CS}$.

Definition 5. Given integers $m, n \geq 0$, an *E(m,n)L grammar* (see [21], [22]) is defined as a quadruple $G = (V, T, P, s)$, where V, T, and s are the total alphabet, the terminal alphabet $T \subseteq V$, and the axiom $s \in V$, respectively. P is a finite set of productions of the form $(u, a, v) \to y$ such that $a \in V$, $u, v, y \in V^*$, $0 \leq |u| \leq m$, and $0 \leq |v| \leq n$. Let $x, y \in V^*$. Then, x *directly derives* y in G, written as $x \Rightarrow_G y$, provided that $x = a_1a_2 \ldots a_k$, $y = y_1y_2 \ldots y_k$, $k \geq 1$, and for all i, $1 \leq i \leq k$, $(a_{i-m} \ldots a_{i-1}, a_i, a_{i+1} \ldots a_{i+n}) \to y_i \in P$. We assume $a_j = \varepsilon$ for all $j \leq 0$ or $j \geq k+1$. In the standard way, $\Rightarrow_G^i$, $\Rightarrow_G^+$, and $\Rightarrow_G^*$ denote the i-fold product of $\Rightarrow_G$, $i \geq 0$, the transitive closure of $\Rightarrow_G$, and the transitive and reflexive closure of $\Rightarrow_G$, respectively. The language of G, $L(G)$, is defined as $L(G) = \{w \in T^* : s \Rightarrow_G^* w\}$. Let $G = (V, T, P, s)$ be an E(0, n)L grammar, $n \geq 0$, and $p = (\varepsilon, A, v) \to y \in P$. We simplify the notation of p so that $p = (A, v) \to y$ throughout this thesis. By *EIL grammars*, we refer to E(m,n)L grammars for all $m, n \geq 0$.

If some grammars define the same language, they are referred to as *equivalent grammars*. This equivalence is central to this thesis because we often discuss how to transform some grammars to some other grammars so that both the original grammars and the transformed grammars are equivalent.

3 Conditions Placed on Derivation Domains

Standardly, the relation of a direct derivation, $\Rightarrow$, is introduced over V^*, where V is the total alphabet of a grammar. Algebraically speaking, $\Rightarrow$ is thus defined over the free monoid whose generators are symbols. In this section, we modify this definition so that we use strings rather than symbols as the generators. More precisely, we introduce this relation over the free monoid generated by a finite set of strings; in symbols, $\Rightarrow$ is defined over W^*, where W is a finite language. This modification represents a very natural context condition: a derivation step is performed on the condition that the rewritten sentential form occurs in W^*. Simultaneously, it results into a strong increase of the generative power of both sequential and parallel context-independent grammars, represented by context-free grammars and E0L grammars, respectively.

Definition 6. A *context-free grammar over word monoid* (a *wm*-grammar for short, see [4], [7]), is a pair (G, W), where $G = (V, T, P, S)$ is a context-free grammar, and W, called the *set of generators*, is a finite language over V. (G, W) is of *degree i*, where i is a natural number, if $y \in W$ implies $|y| \leq i$. (G, W) is said to be *propagating* if $A \rightarrow x \in P$ implies $x \neq \varepsilon$.

The direct derivation $\Rightarrow_{(G,W)}$ on W^* is defined as follows: if $p = A \rightarrow y \in P$, $xAz, xyz \in W^*$ for some $x, z \in V^*$, then xAz *directly derives* xyz, $xAz \Rightarrow_{(G,W)} xyz\ [p]$ in symbols. The language of (G, W), symbolically denoted by $L(G, W)$, is defined as $L(G, W) = \{w \in T^* :\ S \Rightarrow^*_{(G,W)} w\}$.

We denote by **WM** the family of languages generated by *wm*-grammars. The family of languages generated by *wm*-grammars of degree i is denoted by $\mathbf{WM}(i)$. The families of propagating *wm*-grammars of degree i and propagating *wm*-grammars of any degree are denoted by $\textbf{prop-WM}(i)$ and **prop-WM**, respectively.

In [4], Meduna proved the following theorem which establishes the generative power of *wm*-grammars of various degrees:

Theorem 3.

$$\begin{array}{c}\textbf{prop-WM}(1) = \mathbf{WM}(1) = \mathbf{CF} \\ \subset \\ \textbf{prop-WM}(2) = \textbf{prop-WM} = \mathbf{CS} \\ \subset \\ \mathbf{WM}(2) = \mathbf{WM} = \mathbf{RE}.\end{array}$$

Note that the characterization of **RE** can be further improved in such a way that even some reduced versions of *wm*-grammars suffice to generate all the family of recursively enumerable languages (see [7]):

Theorem 4. *Every* $L \in$ **RE** *can be defined by a ten-nonterminal context-free grammar over a word monoid generated by an alphabet and six words of length two.*

Next, we discuss parallel versions of grammars over word monoids. In particular, we define and investigate E0L grammars over word monoids.

Definition 7. An *E0L grammar on word monoid* (a *WME0L grammar* for short, see [6]) is a pair (G, W), where $G = (V, T, P, S)$ is an E0L grammar. The set of generators, W, is a finite language over V. By analogy with *wm-grammars*, (G, W) has *degree* i, where i is a natural number, if every $y \in W$ satisfies $|y| \leq i$. If $A \to x \in P$ implies $x \neq \varepsilon$, (G, W) is said to be propagating. Let $x, y \in W^*$ such that $x = a_1 a_2 \dots a_n$, $y = y_1 y_2 \dots y_n$, $a_i \in V$, $y_i \in V^*$, $1 \leq i \leq n$, $n \geq 0$. If $a_i \to y_i \in P$ for all $i = 1 \dots n$, then x *directly derives* y according to productions $a_1 \to y_1$, $a_2 \to y_2$, …, $a_n \to y_n$, $x \Rightarrow_{(G,W)} y\ [a_1 \to y_1, \dots, a_n \to y_n]$ in symbols. As usual, the list of applied productions is omitted when no confusion arises. The language of (G, W), denoted by $L(G, W)$, is defined in the following way: $L(G, W) = \{w \in T^* : \ S \Rightarrow^*_{(G,W)} w\}$.

By **WME0L**(i), **WMEP0L**(i), **WME0L**, and **WMEP0L**, we denote the families of languages generated by WME0L grammars of degree i, propagating WME0L grammars of degree i, WME0L grammars, and propagating WME0L grammars, respectively. Note that WME0L grammars of degree 2 are called *symbiotic E0L grammars* in [6]. The families of languages generated by symbiotic E0L grammars and propagating symbiotic E0L grammars are denoted by **SE0L** and **SEP0L**; that is, **SE0L** $=$ **WME0L**(2) and **SEP0L** $=$ **WME0L**(2).

Clearly, **WMEP0L**(0) $=$ **WME0L**(0) $= \emptyset$. Recall that for ordinary E0L languages, **EP0L** $=$ **E0L** (see Theorem 2.4 in [23]). Therefore, the following theorem follows immediately from the definitions:

Theorem 5. **WMEP0L**(1) $=$ **WME0L**(1) $=$ **EP0L** $=$ **E0L**.

In [6], Meduna proved that **WMEP0L**(2) $=$ **CS** and **WME0L**(2) $=$ **RE**. Consequently, the following theorem holds:

Theorem 6.

$$\begin{array}{c}\mathbf{CF}\\ \subset\\ \mathbf{WMEP0L}(1) = \mathbf{WME0L}(1) = \mathbf{EP0L} = \mathbf{E0L}\\ \subset\\ \mathbf{WMEP0L}(2) = \mathbf{CS}\\ \subset\\ \mathbf{WME0L}(2) = \mathbf{RE}.\end{array}$$

4 Conditions Placed on the Use of Productions

In this section, we discuss grammars with context conditions represented by strings associated with productions. We distinguish two types of these conditions—*forbidding conditions* and *permitting conditions*. A production is applicable to a sentential form if each of its permitting conditions occurs in the sentential form and any of its forbidding conditions does not. In Section 4.1, we study sequential grammars with context conditions, originally introduced by van der Walt [25] in 1970. Then, in Section 4.2, we introduce and discuss parallel versions of these grammars.

4.1 Sequential Conditional Grammars

Informally, a sequential conditional grammar is an ordinary context-free grammar in which the application of productions is regulated by the permitting and forbidding context conditions. In every derivation step, such a grammar can rewrite only one nonterminal symbol in the given sentential form; that is, it works purely sequentially. Making use of this basic principle, a large number of variants of these grammars have been introduced. In order to unify the notations and definitions, we start with the basic definition of a context-conditional grammar. Then, we investigate some special cases of the context-conditional grammars.

4.1.1 Context-Conditional Grammars

Definition 8. A *context-conditional grammar* is a quadruple, $G = (V, T, P, S)$, where V, T, and S are the total alphabet, the terminal alphabet ($T \subset V$), and the axiom ($S \in V - T$), respectively. P is a finite set of productions of the form $(A \to x, Per, For)$, where $A \in V - T$, $x \in V^*$, and finite sets $Per, For \subseteq V^+$. If $Per \neq \emptyset$ or $For \neq \emptyset$, the production is said to be *conditional*; otherwise, it is called *context-free*. G has *degree* (r, s), where r and s are natural numbers, if for every $(A \to x, Per, For) \in P$, $\max(Per) \leq r$ and $\max(For) \leq s$. If $(A \to x, Per, For) \in P$ implies $x \neq \varepsilon$, G is said to be *propagating*. Let $u, v \in V^*$ and $(A \to x, Per, For) \in P$. Then, u *directly derives* v according to $(A \to x, Per, For)$ in G, denoted by

$$u \Rightarrow_G v \; [(A \to x, Per, For)]$$

provided that for some $u_1, u_2 \in V^*$, the following conditions hold: $u = u_1 A u_2$, $v = u_1 x u_2$, $Per \subseteq \mathrm{sub}(u)$, and $For \cap \mathrm{sub}(u) = \emptyset$. The language of G, denoted by $L(G)$, is defined as

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

The families of languages generated by context-conditional grammars and propagating context-conditional grammars of degree (r, s) are denoted by $\mathbf{CG}(r, s)$ and

prop-CG(r, s), respectively. Furthermore, we define **CG** $= \bigcup_{r=0}^{\infty} \bigcup_{s=0}^{\infty}$ **CG**(r, s) and **prop-CG** $= \bigcup_{r=0}^{\infty} \bigcup_{s=0}^{\infty}$ **prop-CG**(r, s).

Theorem 7.

$$\begin{array}{c}\textbf{prop-CG}(0,0) = \textbf{CG}(0,0) = \textbf{CF} \\ \subset \\ \textbf{prop-CG} = \textbf{CS} \\ \subset \\ \textbf{CG} = \textbf{RE}\end{array}$$

Proof. This proof can be found in the full version of this PhD thesis. ■

4.1.2 Random-Context Grammars

This section discusses three special cases of context-conditional grammars whose conditions are nonterminal symbols, so their degree is not greater than (1,1). Specifically, *random-context grammars*, also known as *permitting grammars*, are of degree (1,0). *Forbidding grammars* are of degree (0,1). Finally, *random-context grammars with appearance checking* are of degree (1,1).

Definition 9. Let $G = (V, T, P, S)$ be a context-conditional grammar. G is called a *random-context grammar with appearance checking* provided that $(A \to x, Per, For) \in P$ implies $Per \subseteq N$ and $For \subseteq N$.

Definition 10. Let $G = (V, T, P, S)$ be a random-context grammar with appearance checking. G is called a *random-context grammar* (an *rc*-grammar for short) or *permitting grammar* provided that every $(A \to x, Per, For) \in P$ satisfies $For = \emptyset$.

Definition 11. Let $G = (V, T, P, S)$ be a random-context grammar with appearance checking. G is called a *forbidding grammar* if every $(A \to x, Per, For) \in P$ satisfies $Per = \emptyset$.

The families of languages defined by random-context grammars, random-context grammars with appearance checking, and forbidding grammars are denoted by **RC**, **RC(ac)**, and **F**, respectively. To indicate that only propagating grammars are considered, we use the prefix **prop-**.

The generative power of random-context grammars is intensively studied in [2] and [18]. These publications present several theorems and lemmas whose results are summarized in the following theorem:

Theorem 8.

$$\textbf{CF} \subset \textbf{prop-RC} \subseteq \textbf{prop-RC(ac)} \subset \textbf{CS},$$

$$\textbf{prop-RC} \subseteq \textbf{RC} \subset \textbf{RC(ac)} = \textbf{RE},$$

$$\textbf{CF} \subset \textbf{ET0L} \subset \textbf{prop-F} \subseteq \textbf{F} \subset \textbf{CS}.$$

4.1.3 Generalized Forbidding Grammars

Generalized forbidding grammars introduced by Meduna in [5] represent a generalized variant of forbidding grammars in which forbidding context conditions are formed by finite languages.

Definition 12. Let $G = (V, T, P, S)$ be a context-conditional grammar. If every $(A \rightarrow x, Per, For)$ satisfies $Per = \emptyset$, then G is said to be a *generalized forbidding grammar* (a *gf*-grammar for short).

The families generated by *gf*-grammars and propagating *gf*-grammars of degree s are denoted by **GF**(s) and **prop-GF**(s), respectively. Furthermore, **GF** $= \bigcup_{s=0}^{\infty}$ **GF**(s) and **prop-GF** $= \bigcup_{s=0}^{\infty}$ **prop-GF**(s).

By analogy with Theorem 7, it is easy to see that *gf*-grammars of degree 0 are ordinary context-free grammars; futhermore, *gf*-grammars of degree 1 are as powerful as forbidding grammars:

Theorem 9. **prop-GF**$(0) =$ **GF**$(0) =$ **CF** $\subset$ **GF**$(1) =$ **F**.

In 1990, Meduna [5] proved that *gf*-grammars of degree 2 generate all the family of recursively enumerable languages:

Theorem 10. **GF**$(2) =$ **GF** $=$ **RE**.

In this thesis, we further improve this result by reducing the number of forbidding productions and nonterminals:

Theorem 11. *Every recursively enumerable language can be defined by a generalized forbidding grammar of degree 2 with no more than 13 forbidding productions and 15 nonterminals.*

Proof. The proof can be found in the full version of this PhD thesis or in [16]. ■

4.1.4 Simple Semi-Conditional Grammars

Simple semi-conditional grammars, a special case of semi-conditional grammars, have been introduced by Meduna and Gopalaratnam in 1994 (see [14]). Informally, a simple semi-conditional grammar is defined as a context-conditional grammar in which every production has no more than one condition.

Definition 13. Let $G = (V, T, P, S)$ be a context-conditional grammar. G is a *simple semi-conditional grammar* (an *ssc*-grammar for short) if $(A \rightarrow x, Per, For) \in P$ implies $|Per| + |For| \leq 1$.

The families of languages generated by *ssc*-grammars and propagating *ssc*-grammars of degree (r, s) are denoted by **SSC**(r, s) and **prop-SSC**(r, s), respectively. Furthermore, **SSC** and **prop-SSC** denote the families of languages generated by *ssc*-grammars and propagating *ssc*-grammars of any degree, respectively.

By Meduna and Gopalaratnam (see [14]), the following relations between the families of languages generated by simple semi-conditional grammars hold:

Theorem 12.

$$\begin{gathered}\mathbf{CF}\\ \subset\\ \textbf{prop-SSC} = \textbf{prop-SSC}(2,1) = \textbf{prop-SSC}(1,2) =\\ \textbf{prop-SC} = \textbf{prop-SC}(2,1) = \textbf{prop-SC}(1,2) = \mathbf{CS}\\ \subset\\ \mathbf{SSC} = \mathbf{SSC}(2,1) = \mathbf{SSC}(1,2) = \mathbf{SC} = \mathbf{SC}(2,1) = \mathbf{SC}(1,2) = \mathbf{RE}.\end{gathered}$$

In this thesis, we demonstrate that there exist a normal form of *ssc*-grammars with a limited number of conditional productions and nonterminals:

Theorem 13. *Every recursively enumerable language can be defined by a simple semi-conditional grammar of degree* $(2, 1)$ *with no more than 12 conditional productions and 13 nonterminals.*

Proof. This proof can be found in the full version of the PhD thesis or in [15]. ■

4.2 Parallel Conditional Grammars

This section studies parallel grammars with permitting and forbidding context conditions. As ET0L grammars represent a very important type of parallel grammars in modern theoretical computer science (see [19], [20], [21], [22], [24]), we base our discussion on these grammars extended by context conditions.

Definition 14. A *context-conditional ET0L grammar* (a *CET0L grammar* for short) is defined as a t+3-tuple, $G = (V, T, P_1, \dots, P_t, S)$, where V, T, and S are the total alphabet, the terminal alphabet ($T \subset V$), and the axiom ($S \in V - T$), respectively. Every P_i, $1 \leq i \leq t$, for some $t \geq 1$, is a finite set of productions of the form $(a \rightarrow x, Per, For)$ with $a \in V$, $x \in V^*$, and $Per, For \subseteq V^+$ are finite languages. A CET0L grammar without erasing productions is said to be *propagating* (a *CEPT0L grammar* for short). G has *degree* (r, s), where r and s are natural numbers, if for every $i = 1, \dots, t$ and $(a \rightarrow x, Per, For) \in P_i$, $\max(Per) \leq r$ and $\max(For) \leq s$. Let $u, v \in V^*$, $u = a_1a_2 \dots a_q$, $v = v_1v_2 \dots v_q$, $q = |u|$, $a_j \in V$, $v_j \in V^*$, and $p_1, p_2, \dots, p_q$ is a sequence of productions $p_j = (a_j \rightarrow v_j, Per_j, For_j) \in P_i$ for all $j = 1, \dots, q$ and some $i \in \{1, \dots, t\}$. If for every p_j, $Per_j \subseteq \mathrm{sub}(u)$

and $For_j \cap \mathrm{sub}(u) = \emptyset$, then u *directly derives* v according to $p_1, p_2, \ldots, p_q$ in G, denoted by

$$u \Rightarrow_G v\ [p_1, p_2, \ldots, p_q].$$

The language of G is defined as

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

If $t = 1$, then G is called a *context-conditional E0L* grammar (a *CE0L grammar* for short). If G is a propagating CE0L grammar, then G is said to be a CEP0L grammar. The families of languages defined by CEPT0L, CET0L, CEP0L, and CE0L grammars of degree (r, s) are denoted by **CEPT0L**(r, s), **CET0L**(r, s), **CEP0L**(r, s), and **CE0L**(r, s), respectively. Set

$$\mathbf{CEPT0L} = \bigcup_{r=0}^{\infty} \bigcup_{s=0}^{\infty} \mathbf{CEPT0L}(r, s), \quad \mathbf{CET0L} = \bigcup_{r=0}^{\infty} \bigcup_{s=0}^{\infty} \mathbf{CET0L}(r, s),$$

$$\mathbf{CEP0L} = \bigcup_{r=0}^{\infty} \bigcup_{s=0}^{\infty} \mathbf{CEP0L}(r, s), \qquad \mathbf{CE0L} = \bigcup_{r=0}^{\infty} \bigcup_{s=0}^{\infty} \mathbf{CE0L}(r, s).$$

Next, we investigate two special cases of context-conditional grammars in detail.

4.2.1 Forbidding ET0L Grammars

This section discusses forbidding ET0L grammars, introduced by Meduna and Švec in [17].

Definition 15. Let $G = (V, T, P_1, \ldots, P_t, S)$ be a CET0L grammar. If every $p = (a \rightarrow x, Per, For) \in P_i$, where $i = 1, \ldots, t$, satisfies $Per = \emptyset$, then G is said to be *forbidding ET0L grammar* (an *FET0L grammar* for short). If G is a propagating FET0L grammar, than G is said to be an *FEPT0L grammar*. If $t = 1$, G is called an *FE0L grammar*. If G is a propagating FE0L grammar, G is called an *FEP0L grammar*.

The families of languages defined by FE0L grammars, FEP0L grammars, FET0L grammars, and FEPT0L grammars of degree s are denoted by **FE0L**(s), **FEP0L**(s), **FET0L**(s), and **FEPT0L**(s), respectively. Moreover,

$$\mathbf{FEPT0L} = \bigcup_{s=0}^{\infty} \mathbf{FEPT0L}(s), \quad \mathbf{FET0L} = \bigcup_{s=0}^{\infty} \mathbf{FET0L}(s),$$

$$\mathbf{FEP0L} = \bigcup_{s=0}^{\infty} \mathbf{FEP0L}(s), \qquad \mathbf{FE0L} = \bigcup_{s=0}^{\infty} \mathbf{FE0L}(s).$$

In this thesis, we establish the following relationships of FET0L language families:

Theorem 14.

$$\begin{array}{c}
\textbf{CF} \\
\subset \\
\textbf{FEP0L}(0) = \textbf{FE0L}(0) = \textbf{EP0L} = \textbf{E0L} \\
\subset \\
\textbf{FEP0L}(1) = \textbf{FEPT0L}(1) = \textbf{FE0L}(1) = \textbf{FET0L}(1) = \\
\textbf{FEPT0L}(0) = \textbf{FET0L}(0) = \textbf{EPT0L} = \textbf{ET0L} \\
\subset \\
\textbf{FEP0L}(2) = \textbf{FEPT0L}(2) = \textbf{FEP0L} = \textbf{FEPT0L} = \textbf{CS} \\
\subset \\
\textbf{FE0L}(2) = \textbf{FET0L}(2) = \textbf{FE0L} = \textbf{FET0L} = \textbf{RE}.
\end{array}$$

Most importantly, we see that FET0L grammars of degree 2 generate the family of recursively enumerable languages and FEPT0L grammars of degree 2 generate the family of context-sensitive languages. Surprisingly, while ordinary ET0L grammars are more powerful than E0L grammars, FET0L(1) and FE0L(1) grammars characterize the same language family.

4.2.2 Simple Semi-Conditional ET0L Grammars

Simple semi-conditional ET0L grammars represent another variant of context-conditional ET0L grammars with restricted sets of context conditions. By analogy with sequential simple semi-conditional grammars (see Section 4.1.4), these grammars are context-conditional ET0L grammars in which every production contains no more than one context condition.

Definition 16. Let $G = (V, T, P_1, \dots, P_t, S)$ be a context-conditional ET0L grammar, for some $t \geq 1$. If for all $p = (a \rightarrow x, Per, For) \in P_i$ for every $i = 1, \dots, t$ holds $|Per| + |For| \leq 1$, G is said to be a *simple semi-conditional ET0L grammar* (*SSC-ET0L grammar* for short). If G is a propagating SSC-ET0L grammar, then G is called an *SSC-EPT0L grammar*. If $t = 1$, then G is called an *SSC-E0L grammar*; if in addition, G is a propagating SSC-E0L grammar, G is said to be an *SSC-EP0L grammar*.

The families of languages generated by SSC-EPT0L grammars of degree (r, s), SSC-ET0L grammars of degree (r, s), SSC-EP0L grammars of degree (r, s), and SSC-E0L grammars of degree (r, s) are denoted by **SSC-EPT0L**(r, s), **SSC-ET0L** (r, s), **SSC-EP0L**(r, s), and **SSC-E0L**(r, s), respectively. The families of languages generated by SSC-EPT0L, SSC-ET0L, SSC-EP0L, and SSC-E0L grammars of any degree are denoted by **SSC-EPT0L**, **SSC-ET0L**, **SSC-EP0L**, and **SSC-E0L** respectively.

The following theorem summarizes the relationships between the language families generated by SSC-ET0L grammars established in the thesis:

Theorem 15.

$$\begin{array}{c}
\mathbf{CF} \\
\subset \\
\textbf{SSC-EP0L}(0,0) = \textbf{SSC-E0L}(0,0) = \mathbf{EP0L} = \mathbf{E0L} \\
\subset \\
\textbf{SSC-EPT0L}(0,0) = \textbf{SSC-ET0L}(0,0) = \mathbf{EPT0L} = \mathbf{ET0L} \\
\subset \\
\textbf{SSC-EP0L}(1,2) = \textbf{SSC-EPT0L}(1,2) = \textbf{SSC-EP0L} = \textbf{SSC-EPT0L} = \mathbf{CS} \\
\subset \\
\textbf{SSC-E0L}(1,2) = \textbf{SSC-ET0L}(1,2) = \textbf{SSC-E0L} = \textbf{SSC-ET0L} = \mathbf{RE}.
\end{array}$$

5 Conditions Placed on the Neighborhood of Rewritten Symbols

In this section, we study grammars with context conditions placed on the neighborhood of rewritten symbols. First, we investigate grammars with context conditions that strictly require a continuous neighborhood of the rewritten symbols. We discuss both sequential and parallel grammars of this kind. Then, we study scattered context grammars, in which rewriting depends on symbols occurring in the sentential form, but these symbols may not form a continuous substring of the sentential form. Rather, these symbols, which are simultaneously rewritten during a single derivation step, may be scattered throughout the sentential form. Since this section contains no results established by the author of this PhD thesis, we only sketch the basic ideas here.

5.1 Continuous Context

This section discusses grammars with continuous context conditions placed on the neighborhood of rewritten symbols—ordinary phrase-structure grammars and EIL grammars. More specifically, it demonstrates that these grammars can be transformed to some special forms in which they generate only sentential forms that have a uniform permutation-based form. These results were established by Meduna in [8] and [11].

5.2 Scattered Context

This section overviews several results regarding the descriptional complexity of scattered context grammars, established by Meduna and Fernau. Perhaps most importantly, Meduna [10] proved that three-nonterminal scattered context grammars generate the family of recursively enumerable languages. Furthermore, Meduna and

Fernau established several normal forms of scattered context grammars (see [12] and [13]) in which only a reduced context-sensitivity and a reduced number of conditional productions and nonterminals suffice to generate all **RE** languages.

6 Grammatical Transformations and Simulations

Classical formal language theory defines equivalent formal models as models that generate the same language. This definition of equivalency plays a crucial role in almost every transformation of formal language models, such as grammars or automata. However, taking a closer look at the transformations of equivalent models, we see that some transformations result in models that generate their languages in a more similar way than others.

Consider such a transformation converting one formal model to another equivalent model. Next, consider a yield sequence generated by the first model and a yield sequence of the second model. If there exists a substitution such that for every step of the first yield sequence, there is a corresponding subsequence of steps in the second yield sequence such that the substitution maps the first and the last string of the subsequence to the strings appearing in the given step of the first yield sequence, we say that the second yield sequence simulates the first one with respect to the given substitution. By a natural generalization of this simulation to all yield sequences of the models, we obtain a simulation-based relationship, reflecting the similarity of the yield sequences of these models. This section provides a formal definition of the above described concept. Then, it introduces a grammatical simulation and, finally, it demonstrates the simulation in terms of Lindenmayer grammars.

6.1 Derivation Simulation

Definition 17. A *string-relation system* is a quadruple $\Psi = (W, \Rightarrow, W_0, W_F)$, where W is a language, $\Rightarrow$ is a binary relation on W, $W_0 \subseteq W$ is a set of *start strings,* and $W_F \subseteq W$ is a set of *final strings.*

Every string, $w \in W$, represents a 0-step string-relation sequence in Ψ. For every $n \geq 1$, a sequence $w_0, w_1, \ldots w_n$, $w_i \in W$, $0 \leq i \leq n$, is an *n-step string-relation sequence,* symbolically written as $w_0 \Rightarrow w_1 \Rightarrow \ldots \Rightarrow w_n$, if for each $0 \leq i \leq n-1$, $w_i \Rightarrow w_{i+1}$.

If there is a string-relation sequence $w_0 \Rightarrow w_1 \Rightarrow \ldots \Rightarrow w_n$, where $n \geq 0$, we write $w_0 \Rightarrow^n w_n$. Furthermore, $w_0 \Rightarrow^* w_n$ means that $w_0 \Rightarrow^n w_n$ for some $n \geq 0$, and $w_0 \Rightarrow^+ w_n$ means that $w_0 \Rightarrow^n w_n$ for some $n \geq 1$. Obviously, from the mathematical point of view, $\Rightarrow^+$ and $\Rightarrow^*$ are the transitive closure of $\Rightarrow$ and the transitive and reflexive closure of $\Rightarrow$, respectively.

Let $\Psi = (W, \Rightarrow, W_0, W_F)$ be a string-relation system. A string-relation sequence in Ψ, $u \Rightarrow^* v$, where $u, v \in W$, is called a *yield sequence,* if $u \in W_0$. If

$u \Rightarrow^* v$ is a yield sequence and $v \in W_F$, $u \Rightarrow^* v$ is *successful*.

Let $D(\Psi)$ and $SD(\Psi)$ denote the set of all yield sequences and all successful yield sequences in Ψ, respectively.

Definition 18. Let $\Psi = (W, \Rightarrow_\Psi, W_0, W_F)$ and $\Omega = (W', \Rightarrow_\Omega, W'_0, W'_F)$ be two string-relation systems, and let σ be a substitution from W' to W. Furthermore, let d be a yield sequence in Ψ of the form $w_0 \Rightarrow_\Psi w_1 \Rightarrow_\Psi \ldots \Rightarrow_\Psi w_{n-1} \Rightarrow_\Psi w_n$, where $w_i \in W$, $0 \leq i \leq n$, for some $n \geq 0$. A yield sequence, h, in Ω *simulates d with respect to σ*, symbolically written as $h \rhd_\sigma d$, if h is of the form $y_0 \Rightarrow_\Omega^{m_1} y_1 \Rightarrow_\Omega^{m_2} \ldots \Rightarrow_\Omega^{m_{n-1}} y_{n-1} \Rightarrow_\Omega^{m_n} y_n$, where $y_j \in W'$, $0 \leq j \leq n$, $m_k \geq 1$, $1 \leq k \leq n$, and $w_i \in \sigma(y_i)$ for all $0 \leq i \leq n$. If, in addition, there exists $m \geq 1$ such that $m_k \leq m$ for each $1 \leq k \leq n$, then h *m-closely simulates d with respect to σ*, symbolically written as $h \rhd_\sigma^m d$.

Definition 19. Let $\Psi = (W, \Rightarrow_\Psi, W_0, W_F)$ and $\Omega = (W', \Rightarrow_\Omega, W'_0, W'_F)$ be two string-relation systems, and let σ be a substitution from W' to W. Let $X \subseteq D(\Psi)$ and $Y \subseteq D(\Omega)$. *Y simulates X with respect to σ*, written as $Y \rhd_\sigma X$, if the following two conditions hold:

1. for every $d \in X$, there is $h \in Y$ such that $h \rhd_\sigma d$;

2. for every $h \in Y$, there is $d \in X$ such that $h \rhd_\sigma d$.

Let m be a positive integer. *Y m-closely simulates X with respect to σ*, $Y \rhd_\sigma^m X$, provided that:

1. for every $d \in X$, there is $h \in Y$ such that $h \rhd_\sigma^m d$;

2. for every $h \in Y$, there is $d \in X$ such that $h \rhd_\sigma^m d$.

Definition 20. Let $\Psi = (W, \Rightarrow_\Psi, W_0, W_F)$ and $\Omega = (W', \Rightarrow_\Omega, W'_0, W'_F)$ be two string-relation systems. If there exists a substitution σ from W' to W such that $D(\Omega) \rhd_\sigma D(\Psi)$ and $SD(\Omega) \rhd_\sigma SD(\Psi)$, then Ω is said to be Ψ's *derivation simulator* and *successful-derivation simulator*, respectively. Furthermore, if there is an integer, $m \geq 1$, such that $D(\Omega) \rhd_\sigma^m D(\Psi)$ and $SD(\Omega) \rhd_\sigma^m SD(\Psi)$, Ω is called an *m-close derivation simulator* and *m-close successful-derivation simulator* of Ψ, respectively. If there exists a homomorphism ρ from W' to W such that $D(\Omega) \rhd_\rho D(\Psi)$, $SD(\Omega) \rhd_\rho SD(\Psi)$, $D(\Omega) \rhd_\rho^m D(\Psi)$, and $SD(\Omega) \rhd_\rho^m SD(\Psi)$, then Ω is Ψ's *homomorphic derivation simulator*, *homomorphic successful-derivation simulator*, *m-close homomorphic derivation simulator* and *m-close homomorphic successful-derivation simulator*, respectively.

6.2 Grammatical Simulation

Consider a typical transformation of a grammar G_1 to another equivalent grammar G_2 in the formal language theory. As a rule, G_2 simulates derivations in G_1 by performing these three phases: (A) *initialization* that produces a string of a desired form by making a few initial steps; (B) *main phase* that actually makes the derivation simulation; (C) *conclusion* that removes various auxiliary symbols.

Phase (B) almost always fulfills a crucial role while the other two phases are usually much less important. Furthermore, phases (A) and (C) usually correspond to no derivation steps in terms of this simulation. As a result, the simulation as a whole is less close than the main phase. Therefore, we next introduce string-relation systems that allow us to formally express phase (B) and, simultaneously, supress the inessential phases (A) and (C).

Making use of the notions introduced in the previous section, we formalize the grammatical simulation in terms of EIL grammars. Let us point out, however, that quite analogically, this simulation can be formalized in terms of any grammatical models.

Definition 21. Let G be an EIL grammar. Let V and T denote G's total and terminal alphabets, respectively. Let $\Rightarrow_G$ be the direct derivation relation in G. For $\Rightarrow_G$ and every $l \geq 0$, set

$$\Delta(\Rightarrow_G, l) = \{x \Rightarrow_G y : \; x \Rightarrow_G y \Rightarrow_G^i w, \; x, y \in V^*, \; w \in T^*, \; i + 1 = l, \; i \geq 0\}.$$

Next, let G_1 be an EIL grammar with total alphabet V_1, terminal alphabet T_1 and axiom $S_1 \in V^*$. Let G_2 be an EIL grammar with total alphabet V_2, terminal alphabet T_2 and axiom $S_2 \in V^*$. Let $\Rightarrow_{G_1}$ and $\Rightarrow_{G_2}$ be the derivation relations of G_1 and G_2, respectively. Let σ be a substitution from V_2 to V_1. G_2 *simulates* G_1 *with respect to* σ, $D(G_2) \rhd_\sigma D(G_1)$ in symbols, if there exists two natural numbers $k, l \geq 0$ so that the following conditions hold:

1. $\Psi_1 = (V_1^*, \Rightarrow_{G_1}, \{S_1\}, T_1^*)$ and $\Psi_2 = (V_2^*, \Rightarrow_{\Psi_2}, W_0, W_F)$ are string-relation systems corresponding to G_1 and G_2, respectively, where $W_0 = \{x \in V_2^* : S_2 \Rightarrow_{G_2}^k x\}$ and $W_F = \{x \in V_2^* : \; x \Rightarrow_{G_2}^l w, \; w \in T_2^*, \; \sigma(w) \subseteq T_1^*\}$;
2. relation $\Rightarrow_{\Psi_2}$ coincides with $\Rightarrow_{G_2} - \Delta(\Rightarrow_{G_2}, l)$;
3. $D(\Psi_2) \rhd_\sigma D(\Psi_1)$.

In case that $SD(\Psi_2) \rhd_\sigma SD(\Psi_1)$, G_2 *simulates successful derivations of* G_1 *with respect to* σ; in symbols, $SD(G_2) \rhd_\sigma SD(G_1)$.

Definition 22. Let G_1 and G_2 be EIL grammars with total alphabets V_1 and V_2, terminal alphabets T_1 and T_2, and axioms S_1 and S_2, respectively. Let σ be a substitution from V_2 to V_1. G_2 *m-closely simulates* G_1 *with respect to* σ if $D(G_2) \rhd_\sigma$

$D(G_1)$ and there exists $m \geq 1$ such that the corresponding string-relation systems Ψ_1 and Ψ_2 satisfy $D(\Psi_2) \rhd_\sigma^m D(\Psi_1)$. In symbols, $D(G_2) \rhd_\sigma^m D(G_1)$.

Analogously, G_2 *m-closely simulates successful derivations of* G_1 *with respect to* σ, denoted by $SD(G_2) \rhd_\sigma^m SD(G_1)$, if $SD(\Psi_2) \rhd_\sigma^m SD(\Psi_1)$ and there exists $m \geq 1$ such that $SD(G_2) \rhd_\sigma^m SD(G_1)$.

Definition 23. Let G_1 and G_2 be two EIL grammars. If there exists a substitution σ such that $D(G_2) \rhd_\sigma D(G_1)$, then G_2 is said to be G_1's *derivation simulator*.

By analogy with Definition 23, the reader can also define *homomorphic m-close derivation simulators* of EIL grammars.

6.3 Close Simulation of E(0,1)L Grammars by Symbiotic E0L Grammars

The above derivation simulation formalism defined in terms of EIL grammars allows us to establish several simulation-based relationships between different variants of Lindenmayer grammars. For example, it can be shown that for every E(0,1)L grammar G, there exist an equivalent symbiotic E0L grammar (G', W) such that (G', W) is a 1-close homomorphic derivation simulator of G:

Theorem 16. *Let* $G = (V, T, P, s)$ *be an E(0,1)L grammar. Then, there exists a symbiotic E0L grammar* (G', W) *and a homomorphism* $\widetilde{\omega}$ *such that* $D(G', W) \rhd_{\widetilde{\omega}}^1 D(G)$ *and* $SD(G', W) \rhd_{\widetilde{\omega}}^1 SD(G)$.

Proof. Formal proof of this statement can be found in the full version of this PhD thesis. ■

7 Applications and Implementation

Although this thesis primarily represents a theoretically oriented treatment, most grammars discussed in the previous chapters have quite realistic applications. Indeed, these grammars are useful to every scientific field that formalizes its results by some strings and studies how these strings are produced from one another under some permitting or, in contrast, forbidding conditions. To illustrate the practical use of the grammars with context conditions, we concentrate our attention on a single application area—biology, which appears of great interest at present.

Case Study 1. Consider a cellular organism in which every cell divides itself into two cells during every single step of a healthy development. However, when a virus infects some cells, all the organism stagnates until it is cured again. During the stagnating period, all the cells just reproduce themselves without producing any new cells. To formalize this development by a suitable simple semi-conditional L grammar, we denote a healthy cell and a virus-infected cell by A and B, respectively, and introduce the simple semi-conditional 0L grammar, $G = (\{A, B\}, P, A)$, where P contains the following productions:

$$\begin{array}{ll} (A \to AA, 0, B), & (B \to B, 0, 0), \\ (A \to A, B, 0), & (B \to A, 0, 0), \\ (A \to B, 0, 0). & \end{array}$$

Figure 1 describes G simulating a healthy development while Figure 2 gives a development with a stagnating period caused by the virus.

In the following case study, we concentrate on a simulation of growing plants generated by parametric 0L grammars extended by context conditions.

Case Study 2. *Parametric 0L grammars* (see [19], [20]) operate on strings of modules called *parametric words*. A *module* is a symbol from an alphabet with an associated sequence of *parameters* belonging to the set of real numbers. Productions of parametric 0L grammars are of the form

$$predecessor\ [\ :\ logical\ expression\ ]\ \to\ successor.$$

The *predecessor* is a module having a sequence of formal parameters instead of real numbers. The *logical expression* is any expression over predecessor's parameters and real numbers. If the logical expression is missing, the logical truth is assumed. The *successor* is a string of modules containing expressions as parameters; for example,

$$A(x)\ :\ x < 7\ \to\ A(x+1)D(1)B(3-x).$$

Such a production *matches* a module in a parametric word provided that the symbol of the rewritten module is the same as the symbol of the predecessor module, both modules have the same number of parameters, and the value for the logical expression is true. Then, the module can be rewritten by the given production.

As usual, a parametric 0L grammar can rewrite a parametric word provided that there exists a matching production for every module that occurs in it. Then, all modules are simultaneously rewritten, and we obtain a new parametric word.

Next, we extend the parametric 0L grammars by permitting context conditions. Each production of a *parametric 0L grammar with permitting conditions* has the form

$$predecessor\ [\ ?\ context\ conditions]\ [\ :\ logical\ expression]\ \to\ successor.$$

where the *predecessor*, the *logical expression*, and the *successor* have the same meaning as in parametric 0L grammars, and *context conditions* are some permitting

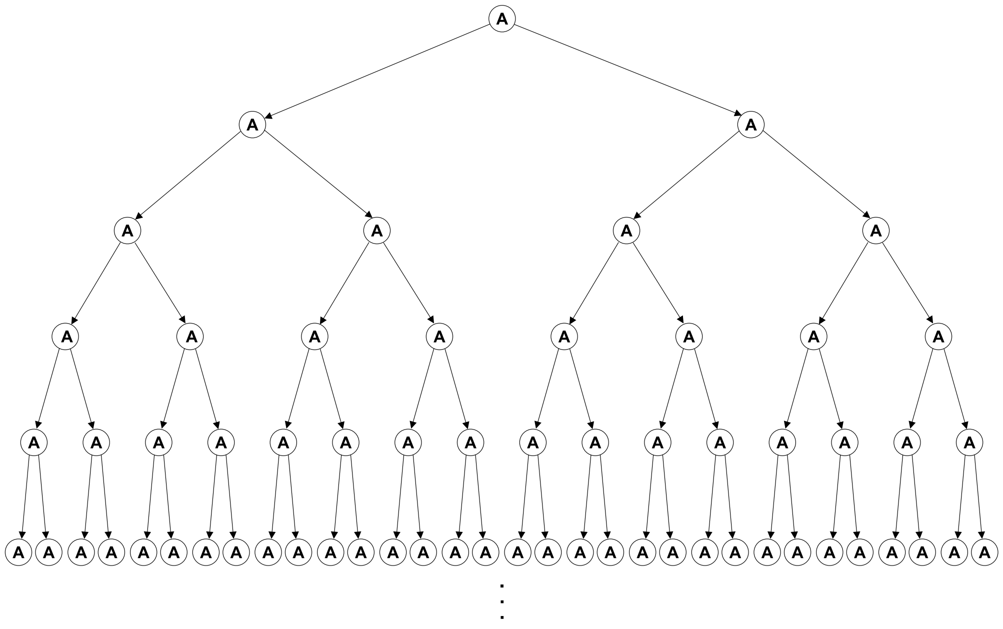
Figure 1: Healthy development.

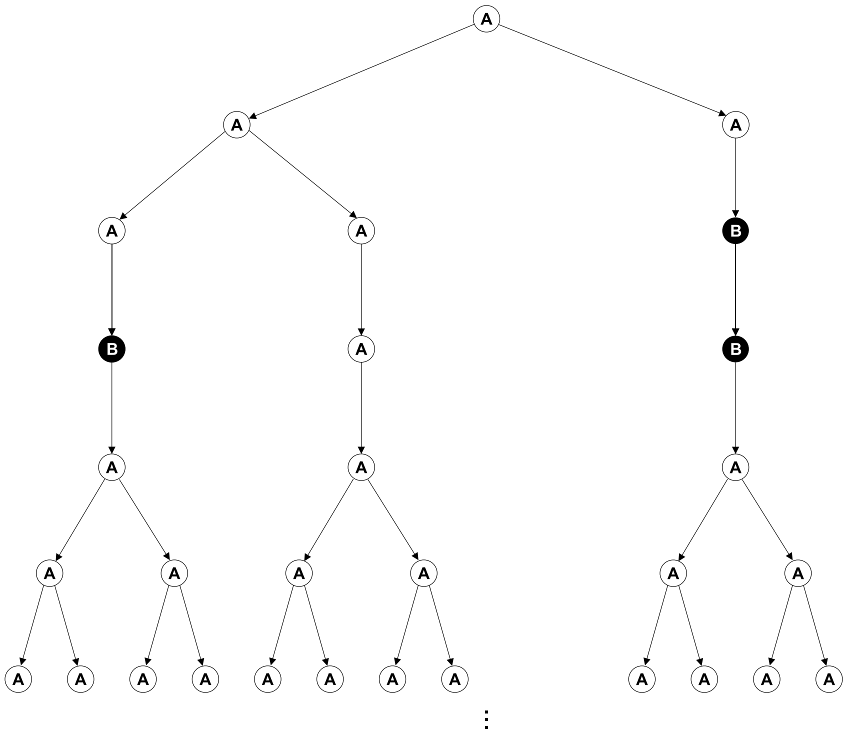
Figure 2: Development with a stagnating period.

context conditions separated by commas. Each condition is a string of modules with formal parameters. For example, consider

$$A(x) \mathrel{?} B(y),\ C(r,z)\ :\ x < y + r\ \rightarrow\ D(x)E(y+r).$$

This production matches a module in a parametric word w provided that the predecessor $A(x)$ matches the rewritten module with respect to the symbol and the number of parameters and there exist modules matching to $B(y)$ and $C(r,z)$ in w such that the value for logical expression $x < y + r$ is true.

Having described the parametric 0L grammars with permitting conditions, we next show how to simulate the development of some plants by using them.

In the nature, developmental processes of multicellular structures are controlled by the quantity of substances exchanged between the modules. In case of plants, the growth depends on the amount of water and minerals absorbed by the roots and carried upwards to the branches. The model of branching structures making use of the resource flow was proposed by Borchert and Honda in [1]. The model is controlled by a *flux* of resources, that starts at the base of the plant and propagates the substances towards the apices. An apex accepts the substances and when the quantity of accumulated resources exceeds a predefined threshold value, the apex bifurcates and initiates a new lateral branch. The distribution of the flux depends on the number of apices that the given branch supports and on the type of the branch—plants usually carry greater amount of resources to straight branches than to lateral branches (see [1] and [19]).

Consider the following model:

$$\begin{array}{lll}
axiom & : & I(1,1,e_{root})\,A(1) \\
p_1 & : & A(id) \mathrel{?} I(id_p,c,e)\ :\ id == id_p\ \wedge\ e \geq e_{th} \\
 & & \rightarrow\ [+(\alpha)\,I(2*id+1,\gamma,0)\,A(2*id+1)]/(\pi)\,I(2*id,1-\gamma,0) \\
 & & \quad A(2*id) \\
p_2 & : & I(id,c,e) \mathrel{?} I(id_p,c_p,e_p)\ :\ id_p == \lfloor id/2 \rfloor \\
 & & \rightarrow\ I(id,c,c*e_p)
\end{array}$$

This L grammar describes a simple plant with a constant resource flow from its roots and with a fixed distribution of the stream between lateral and straight branches. It operates on the following types of modules:

- $I(id,c,e)$ represents an internode with a unique identification number id, a distribution coeficient c, and a flux value e;
- $A(id)$ is an apex growing from the internode with identification number equal to id;
- $+(\phi)$ and $/(\phi)$ rotate the segment orientation by angle ϕ (for more information, consult [19]);

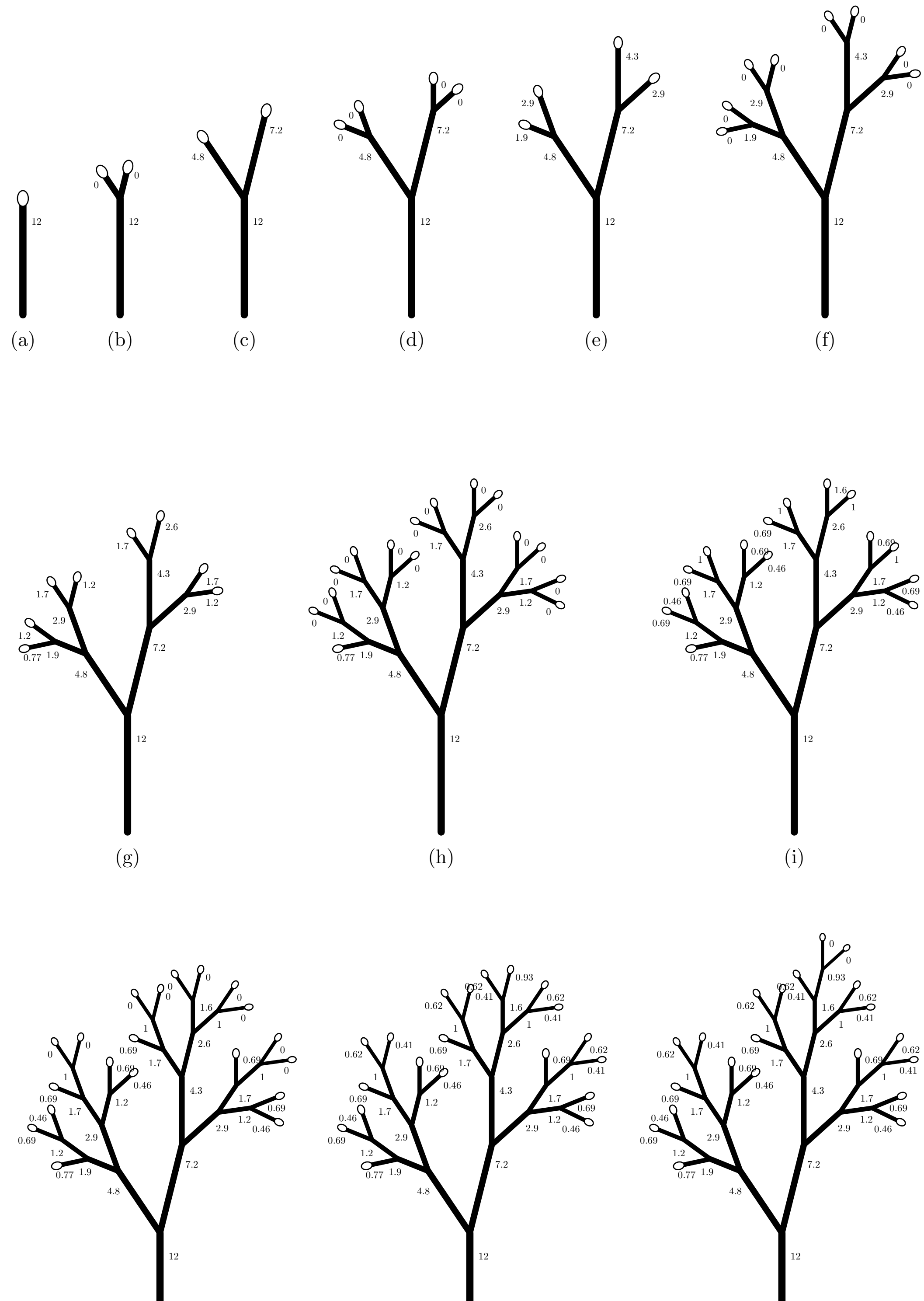
Figure 3: Developmental stages of the plant.

- [and] enclose the sequence of modules describing a lateral branch.

Standardly, we assume that if no production matches a given module $X(x_1, \ldots, x_n)$, the module is rewritten by an implicit production of the form $X(x_1, \ldots, x_n) \rightarrow X(x_1, \ldots, x_n)$; that is, it remains unchanged.

At the beginning, the plant consists of one internode $I(1, 1, e_{root})$ with apex $A(1)$, where e_{root} is a constant flux value provided by roots. The first production, p_1, simulates the bifurcation of an apex. If an internode preceding the apex $A(id)$ reaches a sufficient flux $e \geq e_{th}$, the apex creates two new internodes I terminated by apices A. The lateral internode is of the form $I(2 * id + 1, \gamma, 0)$ and the straight internode is of the form $I(2 * id, 1 - \gamma, 0)$. Clearly, identification numbers of these internodes are unique. Moreover, every child internode can easily calculate the identification number of its parent internode; the parent internode has $id_p = \lfloor id/2 \rfloor$. The coeficient, γ, is a fraction of the parent flux to be directed to the lateral internode. The second production, p_2, controls the resource flow of a given internode. Observe that the permitting condition $I(id_p, c_p, e_p)$ with $id_p = \lfloor id/2 \rfloor$ matches only the parent internode. Thus, p_2 changes the flux value e of $I(id, c, e)$ to $c * e_p$, where e_p is the flux of the parent internode, and c is either γ for lateral internodes or $1 - \gamma$ for straight internodes. Therefore, p_2 simulates the transfer of a given amount of parent's flux into the internode. Figure 3 pictures twelve developmental stages of this plant, with e_{root}, e_{th}, and γ set to 12, 0.9, and 0.4, respectively. The numbers indicate the flow values of internodes.

8 Conclusion

The classical context-dependent grammars, such as context-sensitive and phrase-structure grammars, represent powerful generators of languages. However, their strict context conditions placed on the context surrounding the rewritten symbol during the generation of languages complicate their use both in theory and in practice. Therefore, in this thesis, we discuss a large variety of grammars with much less restrictive context conditions, which are placed on derivation domains, use of productions, or the neighborhood of rewritten symbols. All these grammars use context-independent productions, which obviously significantly simplify the language generation process. Perhaps most importantly, we demonstrate that most of the grammars with alternative context conditions are as powerful as the classical context-dependent grammars. That is, they have the same generative power as the phrase-structure grammars, and if erasing productions are ruled out, they are as powerful as the context-sensitive grammars. From a practical viewpoint, these easy-to-use grammars with flexible context conditions have their important applications in reality as we demonstrate in terms of biology in this thesis as well.

Main goals of this PhD thesis are as follows:

Overview of the Grammars With Context Conditions

Although the language theory have introduced a large number of grammars with context conditions, there exists no monograph dealing with these grammars. Thus, the thesis provide a compact overview of sequential and parallel grammars with context conditions, including their formal definitions and examples.

Reduction of Conditional Grammars

The PhD thesis introduces new results concerning the descriptional complexity of grammars with context conditions. That is, we prove that only a reduced number of conditional productions suffice to generate any recursively enumerable language by generalized forbidding grammars and simple semi-conditional grammars.

Parallel Grammars With Context Conditions

The thesis introduces two parallel variants of conditional grammars—forbidding ET0L grammars and simple semi-conditional ET0L grammars. In both cases, we demonstrate that context conditions significantly increase the generative power of ET0L grammars.

Derivation Simulation

The thesis proposes a general concept formalizing the derivation similarity of rewriting processes of formal language models. Then, it demonstrates its use in terms of Lindenmayer grammars.

Future Research

This thesis opens several new research areas in the language theory. For instance, it introduces and discusses some conditional variants of ET0L grammars. However, there exist a number of parallel and semi-parallel grammars whose variants with context conditions have not been studied yet. Because the computational parallelism plays a crucial role in today's computer science, parallel versions of conditional grammars clearly represent an important contribution to the research of parallel models.

The concept of the derivation simulation gives rise to a number of questions as well. Indeed, consider any transformation between two different types of grammars. Then, it is possible to discuss the following questions: Does the transformation result in a grammar that simulates the input grammar? If so, what degree of closeness the transformation guarantees? Is it possible to find another transformation with better simulation properties?

Finally, there is a number of practical applications in which the theory of grammars with context conditions might be applied in the future. These applications range from the language theory, compilers and parallel computations to the simulation of biological structures, molecular biology and genetics.

References

[1] R. Borchert and H. Honda. Control of development in the bifurcating branch system of *Tabebuia Rosea*: A computer simulation. *Botanical Gazette*, 145(2):184–195, 1984.

[2] J. Dassow and Gh. Paun. *Regulated Rewriting in Formal Language Theory*. Akademie-Verlag, Berlin, 1989.

[3] J. E. Hopcroft and J. D. Ullman. *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley, Reading, Massachusetts, 2nd edition, 1979.

[4] A. Meduna. Context-free derivations on word monoids. *Acta Informatica*, 27:781–786, 1990.

[5] A. Meduna. Generalized forbidding grammars. *International Journal of Computer Mathematics*, 36:31–38, 1990.

[6] A. Meduna. Symbiotic E0L systems. *Acta Cybernetica*, 10:165–172, 1992.

[7] A. Meduna. Syntactic complexity of context-free grammars over word monoids. *Acta Informatica*, 33:457–462, 1996.

[8] A. Meduna. Uniform rewriting based on permutations. *International Journal of Computer Mathematics*, 69:57–74, 1998.

[9] A. Meduna. *Automata and Languages: Theory and Applications*. Springer, London, 2000.

[10] A. Meduna. Generative power of three-nonterminal scattered context grammars. *Theoretical Computer Science*, 246:276–284, 2000.

[11] A. Meduna. Uniform generation of languages by scattered context grammars. *Fundamenta Informaticae*, 44:231–235, 2001.

[12] A. Meduna and H. Fernau. On the degree of scattered context-sensitivity. *Theoretical Computer Science*, 290:2121–2124, 2003.

[13] 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.

[14] A. Meduna and A. Gopalaratnam. On semi-conditional grammars with productions having either forbidding or permitting conditions. *Acta Cybernetica*, 11:307–323, 1994.

[15] 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.

[16] A. Meduna and M. Švec. Descriptional complexity of generalized forbidding grammars. *International Journal of Computer Mathematics*, 80(1):11–17, 2003.

[17] A. Meduna and M. Švec. Forbidding ET0L grammars. *Theoretical Computer Science*, 306:449–469, 2003.

[18] M. Penttonen. ET0L-grammars and N-grammars. *Information Processing Letters*, 4:11–13, 1975.

[19] P. Prusinkiewicz, M. Hammel, J. Hanan, and R. Měch. L-systems: From the theory to visual models of plants. In M. T. Michalewicz, editor, *Proceedings of the 2nd CSIRO Symposium on Computational Challenges in Life Sciences*, Collingwood, Victoria, Australia, 1996. CSIRO Publishing.

[20] P. Prusinkiewicz and A. Lindenmayer. *The Algorithmic Beauty of Plants*. Springer-Verlag, New York, 1990.

[21] G. Rozenberg and A. Salomaa. *The Mathematical Theory of L Systems*. Academic Press, New York, 1980.

[22] G. Rozenberg and A. Salomaa. *The Book of L*. Springer-Verlag, Berlin, 1986.

[23] G. Rozenberg and A. Salomaa. *Handbook of Formal Languages*, volume 1–3. Springer, Berlin, 1997.

[24] S. H. von Solms. Modelling the growth of simple biological organisms using formal language theory. *Manuscript*.

[25] A. P. J. van der Walt. Random context grammars. In *Proceedings of the Symposium on Formal Languages*, 1970.

Author's Curriculum Vitae

Name:	Martin Švec
Born:	Brno, Czech Republic, 1979
Education:	Brno University of Technology, Czech Republic, Faculty of Electrical Engineering and Computer Science (1997-2001), Faculty of Information Technology (2002), M.Sc. in Computer Science and Engineering (2002), Ph.D. student at the Faculty of Information Technology (2002-2005).
Teaching:	Theoretical exercises of M.Sc. courses *Principles of Compiler Design* and *Formal Languages and Compilers*.
Publications:	Co-author of three papers published in international journals, co-author of a monograph discussing grammars with context conditions (John Wiley & Sons, 2005).