BRNO UNIVERSITY OF TECHNOLOGY
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

SYNCHRONOUS FORMAL SYSTEMS BASED ON GRAMMARS AND TRANSDUCERS

SYNCHRONNÍ FORMÁLNÍ SYSTÉMY ZALOŽENÉ NA GRAMATIKÁCH A PŘEVODNÍCÍCH

PHD THESIS EXTENDED ABSTRACT
TEZE DISERTAČNÍ PRÁCE

AUTHOR PETR HORÁČEK
AUTOR PRÁCE

SUPERVISOR Prof. RNDr. ALEXANDER MEDUNA, CSc.
VEDOUCÍ PRÁCE

BRNO 2014

Abstract
This doctoral thesis studies synchronous formal systems based on grammars and transducers, investigating both theoretical properties and practical application perspectives. It introduces new concepts and definitions building upon the well-known principles of regulated rewriting and synchronization. An alternate approach to synchronization of context-free grammars is proposed, based on linked rules. This principle is extended to regulated grammars such as scattered context grammars and matrix grammars. Moreover, based on a similar principle, a new type of transducer called the rule-restricted transducer is introduced as a system consisting of a finite automaton and context-free grammar. New theoretical results regarding the generative and accepting power are presented. The last part of the thesis studies linguistically-oriented application perspectives, focusing on natural language translation. The main advantages of the new models are discussed and compared, using select case studies from Czech, English, and Japanese to illustrate.

Abstrakt
Tato disertační práce studuje synchronní formální systémy založené na gramatikách a převodnících a zkoumá jak jejich teoretické vlastnosti, tak i perspektivy praktických aplikací. Práce představuje nové koncepty a definice vycházející ze známých principů řízeného přepisování a synchronizace. Navrhuje alternativní způsob synchronizace bezkontextových gramatik, založený na propojení pravidel. Tento princip rozšiřuje také na řízené gramatiky, konkrétně gramatiky s rozptýleným kontextem a maticové gramatiky. Dále je představen na podobném principu založený nový druh převodníku, tzv. pravidlově omezený převodník. Jedná se o systém složený z konečného automatu a bezkontextové gramatiky. Práce prezentuje nové teoretické výsledky ohledně generativní a přijímající síly. Poslední část práce zkoumá možnosti lingvisticky orientovaných aplikací se zameřením na překlad přirozeného jazyka. Diskutuje a srovnává hlavní výhody nových modelů s využitím vybraných případových studií z českého, anglického a japonského jazyka pro ilustraci.

Keywords
formal systems, grammars, transducers, regulated rewriting, synchronization, natural language syntax, natural language translation

Klíčová slova
formální systémy, gramatiky, převodníky, řízené přepisování, synchronizace, syntaxe přirozeného jazyka, překlad přirozeného jazyka

Citation
Petr Horáček: Synchronous Formal Systems Based on Grammars and Transducers, PhD thesis, Brno, FIT BUT, 2014

Citace
Petr Horáček: Synchronous Formal Systems Based on Grammars and Transducers, disertační práce, Brno, FIT VUT v Brně, 2014

Contents

Chapter 1

Introduction

Formal language theory is an essential part of theoretical computer science. It defines and studies languages as sets of strings (words, sentences), which are finite sequences of symbols. This definition covers natural languages (e.g. Czech, English, or Japanese) as well as artificial languages (such as programming languages).

To describe languages mathematically, formal language theory studies models which define them. Many of these models are based on rewriting systems—that is, formal systems which gradually change strings by rewriting some of their symbols in each step, according to a given set of rules. Most rewriting systems fall into one of the two basic categories: generative language models (generally known as grammars), and accepting language models (generally known as automata). A generative model defines a language by generating all strings of this language. In other words, a string belongs to this language if and only if it can be generated by the model. An accepting model analyzes a string and either accepts, or rejects it. The language defined by the accepting model is the set of all strings which the model accepts.

The applications of formal language theory are found in many scientific disciplines. It provides mathematical background primarily in areas that deal with languages themselves (linguistics, programming language theory etc.) but there are also other topics that can be formalized as languages (e.g. DNA and RNA sequences in biology).

Of particular interest to our work is the area of computational linguistics. Specifically, we focus on formal description of natural language syntax and its transformations. We study the application perspectives of known formal models and introduce new related concepts and definitions. We also study the theoretical properties of the models and present new results.

1.1 Motivation

Natural language processing is a field of theoretical informatics and linguistics and is concerned with the interactions between computers and human (natural) languages. It is defined as a theoretically motivated range of computational techniques for analyzing and representing naturally occurring texts (which means any language) at one or more levels of linguistic analysis for the purpose of achieving human-like language processing for a range of tasks or applications (according to [4]).

The history goes back to the the late 1940s, when there was an effort to understand and formally describe the syntax of natural languages. A big step forward was the publishing of the book called Syntactic Structures, by Noam Chomsky, introducing the idea of generative grammar.

At first, computer processing of natural languages was in interest of artificial inteligence as a part of human-computer interaction. Subsequently, it split into two separate disciplines. Today, natural language processing studies many other aspects of natural languages besides their syntax (such as morphology or semantics). This discipline is focused mainly on practical applications. Some of the most frequent tasks are information retrieval, information extraction, question answering, summarization, and machine translation, and in broader scope, we can even include tasks as speech recognition and speech synthesis.

The second discipline encompasses a set of formalisms, which are, in general, known as formal language theory. Formal language theory is considered a part of theoretical computer science, and it focuses mainly on theoretical studies of various formal models and their properties. Its applications are now found in many other areas besides computational linguistics.

One of the major trends in formal language theory is regulated rewriting. This concept was introduced already in the 1960s, as the models of the now traditional Chomsky hierarchy have been found unsatisfactory for certain practical applications. For example, it has been argued that some linguistic phenomena could not be described by context-free grammars, while context-sensitive and unrestricted grammars were inefficient for practical use (because of the complexity of parsing). Because of this, ways to increase the power of context-free grammars—while retaining their practical applicability—were investigated.

Regulated rewriting essentially means that we take a certain known formal model (usually a context-free grammar, for reasons mentioned above) and in some way regulate (hence the name) the way in which it generates (or, in the case of automata, accepts) sentences. This can be done by adding some mathematically simple mechanism that controls the use of rules (such as in programmed grammars), or by changing the form of rules themselves (as, for example, in scattered context grammars). Thus, the expressive power is increased by limiting available derivations (or computations).

The purpose of our work is twofold. From a theoretical point of view, we contribute to the study of formal language theory by introducing new formal models and investigating their properties. Rather than trying to create completely new formalisms from scratch, we establish the new models as generalizations, extensions, or modifications of well-known and well-studied formal models (such as context-free grammars and finite automata) and principles (such as regulated rewriting and synchronization).

In [22], we have presented an alternate approach to synchronization, based on linking rules instead of nonterminals. In this fashion, we have extended the principle to models with regulated rewriting, specifically matrix grammars and scattered context grammars. We have continued with further theoretical study of synchronous grammars based on linked rules, and particularly of synchronous versions of regulated grammars, in [20] and [23].

In [6], we have introduced a new type of transducer, the rule-restricted automaton-grammar transducer, as a system consisting of a finite automaton, which is used to read an input string, and a context-free grammar, which simultaneously produces a corresponding output string. Also in [6], we have investigated the theoretical properties—namely, the generative and accepting power—of this new system and its variants.

For an overview of our new results, see the following Section 1.2, specifically the parts describing Chapter 2 and Chapter 3 for results concerning synchronous grammars and transducers, respectively.

Meanwhile, from a more practical viewpoint, we investigate how some of the well-known and well-studied models from formal language theory can be adapted or extended for applications in natural language processing. In other words, the ideas and concepts behind the new formal models mentioned above are motivated by the possibility of their linguistic applications.

Inspired by such works as [32], where the authors discuss linguistically-oriented applications of scattered context grammars (using examples from the English language), we explore similar application perspectives of other regulated formal models as well. In [18], we have discussed potential applications of matrix grammars in the description of the Japanese syntax. Subsequently, we have been focusing on translation of natural languages.

Machine translation is one of the major tasks in natural language processing. With increasing availability of large corpora, corpus-based systems became favoured over rule-based, using statistical methods and machine-learning techniques. They mostly rely on formal models that represent local information only, such as n-gram models. However, recently, there have been attempts to improve results by incorporating syntactic information into such systems (see [26], [42], or [5]).

To do so, we need formal models that can describe syntactic structures and their transformations. Based on the principles of synchronous grammars (see [8]), we have proposed synchronous versions of some regulated grammars, such as matrix grammars (see [12]) and scattered context grammars (see [32]). We first introduced the idea in [19], and further elaborated upon it in [22]. Revised definitions, a study of theoretical properties, and a further discussion of linguistically-oriented application perspectives can be found in [23]; applications in particular are also investigated in [21].

Other type of models we can use are transducers (see [2]). Unlike synchronous grammars, which generate a pair of sentences in one derivation and thus define translation, transducers take a given input sentence and transform it into a corresponding output sentence. Frequently, these transducers consist of several components, including various automata and grammars, some of which read their input strings while others produce their output strings (see [15] or [36]). In [6], we have introduced the rule-restricted automaton-grammar transducer and its variants, and discussed its advantages for natural language translation, illustrated by examples from Czech, English, and Japanese.

1.2 Thesis Organization

This doctoral thesis is divided into three parts and seven chapters, organized as follows.

1.2.1 Introduction

The first chapter introduces the topic of our work and presents the motivation behind it. It also describes the structure of this document and provides an overview of its contents.

Following this introductory chapter, Chapter 2 provides the mathematical background required for understanding of the topics discussed in this work. First, we summarize the well-known essential concepts and definitions from set theory, such as sets and relations. Subsequently, we use these notions to present an introduction to formal language theory. We give formal definitons of concepts such as alphabet, string, and language. We also introduce formal models that define languages, namely grammars and automata. We define different types of languages and present the resulting hierarchy of the respective language classes. Finally, we describe and formally define several models related to the concept of regulated rewriting.

In Chapter 3, we present a brief introduction to computational linguistics. The first section of this chapter provides an overview of select formal models related to natural language processing. We discuss both models of historical and practical importance. Transformational grammars, augmented transition networks and generalized phrase structure grammars are examples of the former category, while the latter includes head-driven phrase structure grammars, lexical functional grammars, and lexicalized tree-adjoining grammars. We also mention probabilistic context-free grammar as an example of a formal model used in statistical natural language processing.

We present the basic concept of dependency grammars as well. Although for the most part our work does not deal with dependency grammars, they certainly deserve a mention as an important alternative to phrase structure grammars, which is is also often used in practice. Moreover, we sometimes use the notion of dependency (and some related notions, particularly nonprojectivity, the crossing of dependencies) when discussing application perspectives of our formal models (Chapter 6).

Finally, we consider the application of some traditional models from formal language theory as an alternative. We focus on models with regulated rewriting. In particular, we discuss linguistically-oriented application perspectives of scattered context grammars and their variant, transformational scattered context grammars.

The second section of Chapter 3 introduces the area of machine translation, which is one of major tasks of natural language processing. First, we briefly review the historical development and classification of translation systems. Subsequently, we provide a summary of recent trends in this area, and we show how our work relates to them.

1.2.2 Synchronous Formal Systems

The second part of this thesis consists of two theoretically oriented chapters. These chapters contain both informal explanations and formal definitions of the new models, and present the related theoretical results that we have established.

More specifically, Chapter 4 deals with synchronous grammars. First, we briefly recall the well-known synchronous context-free grammars. We then introduce the notion of new synchronous grammars as systems consisting of two context-free grammars with linked rules instead of linked nonterminals. This allows us to naturally extend the principle of synchronization beyond context-free grammars. We present synchronous versions of some regulated grammars, namely scattered-context grammars and matrix grammars.

Further, we study theoretical properties of these grammars. Specifically, we investigate their generative power and achieve the following three main results. First, if we synchronize context-free grammars by linking rules as proposed and defined in this chapter, we obtain generative power coinciding with the power of matrix grammars. Consequently, we significantly increase the power in this way because the traditional synchronous CFGs only generate the family of context-free languages. Second, perhaps unsurprisingly, the class of languages defined by synchronous scattered context grammars equals the class of recursively enumerable languages. Finally, we show that if we synchronize matrix grammars by linking matrices, we obtain no increase in power. That is, synchronous matrix grammars have the same generative power as matrix grammars.

Chapter 5 introduces a new type of transducer, referred to as rule-restricted automaton-grammar transducer, based upon a finite automaton and a context-free grammar. A restriction set controls the computation. It defines which rules can be simultaneously used by the automaton and by the grammar. We discuss the power of this system working in an ordinary way as well as in a leftmost way (more precisely, the context-free grammar is restricted to leftmost derivation). In addition, we introduce an appearance checking, which allows us to check whether some symbols are present in the rewritten string, and we investigate its effect on the power.

We achieve the following main results. First, we show that the generative power of rule-restricted transducers is equal to the generative power of matrix grammars. Second, the accepting power coincides with the power of partially blind multi-counter automata. Third, under the context-free-grammar leftmost restriction, the accepting and generative power of these systems coincides with the power of context-free grammars. On the other hand, when an appearance checking is introduced into these systems, the system can accept and generate all recursively enumerable languages.

1.2.3 Application Perspectives and Final Remarks

In the final part of this thesis, we consider the newly introduced models from a more practical viewpoint. Specifically, Chapter 6 explores their application perspectives with particular focus on natural language translation. We discuss and compare their main advantages illustrating them by examples from Czech, English, and Japanese.

One of the main advantages of both types of presented models is their power, as they are able to describe even some non-context-free structures. The new synchronous grammars also provide high flexibility, allowing for elegant and efficient description of language features. On the other hand, rule-restricted transducers are based on a simple, straightforward principle, which can be an advantage for practical implementation.

The concluding Chapter 7 summarizes all achieved results. In particular, using a graphical representation, we show how our new results relate to a known hierarchy of language classes. We also discuss further research prospects, both in theoretical and practical direction.

1.3 Extended Abstract Organization

In this extended abstract of the doctoral thesis, we assume that the reader is familiar with the fundamental concepts and models of modern formal language theory (see [30] or [39]) and natural language processing (see [4] or [35]). We therefore omit Chapters 2 and 3 of the full thesis, which cover these topics.

Chapters 2 and 3 of this extended abstract are abridged versions of Chapters 4 and 5 of the thesis (the theoretically-oriented Part II, presenting new definitions and results for synchronous grammars and transducers).

Chapters 4 and 5 correspond to Part III of the thesis, specifically to Chapter 6 dealing with linguistically-oriented application perspectives and Chapter 7 concluding the thesis, respectively.

Chapter 2

Synchronous Systems Based on Grammars

In essence, synchronous grammars are grammars or grammar systems that generate pairs of sentences in one derivation, instead of single sentences (as for example in CFGs). In this way, they allow us to describe translations. That is, in each pair, the first string is a sentence of the source language, and the second string is a corresponding sentence of the target language.

Although the term synchronous context-free grammar (SCFG for short) is relatively recent, the essential principle was introduced already in the late 1960s in syntax-directed translation schemata [3] and syntax-directed transduction grammars [27]. These models were originally developed as formal background for compilers of programming languages. Subsequently, synchronous grammars have been succesfully used natural language processing as well, particularly in machine translation (see Section 3.2 of the full thesis).

Informally, we can see SCFG (see [8] or [9]) as a modification of CFG where every rule has two right-hand sides, the first of which is applied to the input sentential form (source), and the second to the output sentential form (target). Nonterminals are linked, which means that in each derivation step, we rewrite both the selected nonterminal symbol in the input sentential form and its appropriate counterpart in the output sentential form.

Example 2.1. The following two rules are a fragment of a synchronous CFG which transforms arithmetic expressions from infix notation (e.g. $3 \times 5 + 4$) to postfix notation (e.g. $3\,5 \times 4\,+$). E, F, and T are nonterminals, $+$ and $\times$ are terminals, E is the start symbol.

$$\begin{aligned} 1 &: \ \mathrm{E} \rightarrow \mathrm{E}_{\boxed{1}} + \mathrm{T}_{\boxed{2}},\ \mathrm{E}_{\boxed{1}}\ \mathrm{T}_{\boxed{2}}\ + \\ 2 &: \ \mathrm{T} \rightarrow \mathrm{T}_{\boxed{1}} \times \mathrm{F}_{\boxed{2}},\ \mathrm{T}_{\boxed{1}}\ \mathrm{F}_{\boxed{2}}\ \times \end{aligned}$$

A derivation using these rules may look like this:

$$\begin{aligned} (\mathrm{E}_{\boxed{42}}, \mathrm{E}_{\boxed{42}}) &\Rightarrow (\mathrm{E}_{\boxed{43}} + \mathrm{T}_{\boxed{44}}, \mathrm{E}_{\boxed{43}}\ \mathrm{T}_{\boxed{44}}\ +)\ [1] \\ &\Rightarrow (\mathrm{E}_{\boxed{45}} + \mathrm{T}_{\boxed{46}} + \mathrm{T}_{\boxed{47}}, \mathrm{E}_{\boxed{45}}\ \mathrm{T}_{\boxed{46}} + \mathrm{T}_{\boxed{47}}\ +)\ [1] \\ &\Rightarrow (\mathrm{E}_{\boxed{45}} + \mathrm{T}_{\boxed{48}} \times \mathrm{F}_{\boxed{49}} + \mathrm{T}_{\boxed{47}}, \mathrm{E}_{\boxed{45}}\ \mathrm{T}_{\boxed{48}}\ \mathrm{F}_{\boxed{49}} \times + \mathrm{T}_{\boxed{47}}\ +)\ [2] \end{aligned}$$

The boxed numbers are used to denote linked nonterminals. That is, two nonterminals are linked if they have the same number (e.g. $\boxed{1}$ and $\boxed{1}$). Every derivation in a SCFG starts with a pair of linked nonterminals, such as $(\mathrm{E}_{\boxed{42}}, \mathrm{E}_{\boxed{42}})$ here (the starting number 42 is chosen arbitrarily). Whenever we make a derivation step, we assign new, unique numbers to each newly introduced pair of linked nonterminals, as seen in the derivation example above.

In each derivation step, we can only rewrite linked nonterminals (nonterminals sharing the same boxed number). Note that when applying rule 2 above, we rewrite the first occurence of T in both sentential forms, which is allowed, as it is $\mathrm{T}_{\boxed{46}}$ in both cases. We could also choose to rewrite the second occurence in both sentential forms ($\mathrm{T}_{\boxed{47}}$). However, we cannot choose the first T in one sentential form and the second T in the other, because the assigned numbers do not correspond ($\mathrm{T}_{\boxed{46}}$ and $\mathrm{T}_{\boxed{47}}$), and thus the two nonterminals are not considered linked.

The original ideas, concepts, definitions, and theoretical results presented in this chapter were first published in [22] and [23].

2.1 Rule-Synchronized Context-Free Grammar

In [22] and [23], we have proposed synchronization based on linked rules instead of nonterminals. Informally, such synchronous grammar is a system of two grammars, G_I and G_O, in which the corresponding rules share labels. For example, if we apply rule labelled 1 in the input grammar G_I, we also have to apply rule labelled 1 in the output grammar G_O, and this makes for a single derivation step in the synchronous grammar. In other words, the input and output sentence have the same parse (a sequence of rules applied in a derivation, denoted by their labels).

Example 2.2. Rules (G_I on the left, G_O on the right):

$$\begin{array}{ll} 1: \mathrm{E} \rightarrow \mathrm{E} + \mathrm{T} & 1: \mathrm{E} \rightarrow \mathrm{E}\ \mathrm{T} + \\ 2: \mathrm{T} \rightarrow \mathrm{T} \times \mathrm{F} & 2: \mathrm{T} \rightarrow \mathrm{T}\ \mathrm{F} \times \end{array}$$

An example of a derivation using these rules in G_I follows.

$$\mathrm{E} \Rightarrow \mathrm{E} + \mathrm{T}\ [1] \Rightarrow \mathrm{E} + \mathrm{T} \times \mathrm{F}\ [2]$$

A corresponding derivation in G_O is:

$$\mathrm{E} \Rightarrow \mathrm{E}\ \mathrm{T} +\ [1] \Rightarrow \mathrm{E}\ \mathrm{T}\ \mathrm{F} \times +\ [2]$$

The parse is $(1, 2)$.

However, note that we place no restriction on the linked rules. For instance, unlike in synchronous CFGs, we do not have to rewrite the same nonterminal in both sentential forms in one derivation step. Both the right-hand sides and the left-hand sides of linked rules may be completely different, for example:

$$3: \mathrm{A} \rightarrow \mathrm{B\ a\ C} \qquad 3: \mathrm{P} \rightarrow \mathrm{Q\ B\ R\ b\ d}$$

In other words, rule-synchronized CFGs can be seen as a generalization of the traditional synchronous CFGs, as the latter can be defined as special case of rule-synchronized CFGs, where each two linked rules have the same left-hand side (that is, they rewrite the same nonterminal).

Formally, we define a rule-synchronized CFG as follows.

Definition 2.1 (Rule-synchronized CFG)**.** A *rule-synchronized CFG* (RSCFG for short) H is a quintuple $H = (G_I, G_O, \Psi, \varphi_I, \varphi_O)$, where

- $G_I = (N_I, T_I, P_I, S_I)$ and $G_O = (N_O, T_O, P_O, S_O)$ are CFGs,
- Ψ is a set of *rule labels*, and
- φ_I is a function from Ψ to P_I and φ_O is a function from Ψ to P_O.

We say that two rules $p_I \in P_I$ and $p_O \in P_O$ are linked, if and only if there is some label $p \in \Psi$ such that $\varphi_I(p) = p_I$ and $\varphi_O(p) = p_O$. That is, each two linked rules share the same label.

We use the following notation (presented for input grammar G_I, analogous for output grammar G_O):

$p : A_I \to x_I$ where $p \in \Psi, A_I \to x_I \in P_I$	$\varphi_I(p) = A_I \to x_I$
$x_I \Rightarrow_{G_I} y_I\, [p]$ where $x_I, y_I \in (N \cup T)^*, p \in \Psi$	derivation step in G_I applying rule $\varphi_I(p)$
$x_I \Rightarrow^n_{G_I} y_I\, [p_1 \dots p_n]$ where $x_I, y_I \in (N \cup T)^*, p_i \in \Psi$ for $1 \le i \le n$	derivation in G_I applying rules $\varphi_I(p_1) \dots \varphi_I(p_n)$

Definition 2.2 (Translation in RSCFG)**.** Let $H = (G_I, G_O, \Psi, \varphi_I, \varphi_O)$ be an RSCFG. The *translation defined by* H, denoted by $T(H)$, is the set of pairs of sentences, which is defined as

$$\begin{aligned} T(H) = \{(w_I, w_O) \quad : \quad & w_I \in T_I^*, w_O \in T_O^*, \\ & S_I \Rightarrow^*_{G_I} w_I\, [\alpha], S_O \Rightarrow^*_{G_O} w_O\, [\alpha] \text{ for some } \alpha \in \Psi^*\}. \end{aligned}$$

Originally [22], we considered RSCFG only as a variant of synchronous CFG. However, there is in fact a significant difference. While the latter does not increase the generative power over CFG, RSCFG does, as is shown in the next subsection.

2.1.1 Generative Power

Synchronous grammars define translations—that is, sets of pairs of sentences. To be able to compare their generative power with well-known models such as CFGs, which define languages, we can consider their input and output language separately.

Definition 2.3 (Input and output language)**.** Let H be an RSCFG. Then, we define

- the *input language* of H, denoted by $L_I(H)$, as $L_I(H) = \{w_I : (w_I, w_O) \in T(H)\}$, and
- the *output language* of H, denoted by $L_O(H)$, as $L_O(H) = \{w_O : (w_I, w_O) \in T(H)\}$.

Example 2.3. Consider an RSCFG $H = (G_I, G_O, \Psi, \varphi_I, \varphi_O)$ with the following rules (nonterminals are in capitals, linked rules share the same label, S_I and S_O are the start symbols of G_I and G_O, respectively):

G_I				G_O			
1 :	S_I	$\to$	ABC	1 :	S_O	$\to$	A
2 :	A	$\to$	aA	2 :	A	$\to$	B
3 :	B	$\to$	bB	3 :	B	$\to$	C
4 :	C	$\to$	cC	4 :	C	$\to$	A
5 :	A	$\to$	ε	5 :	A	$\to$	B'
6 :	B	$\to$	ε	6 :	B'	$\to$	C'
7 :	C	$\to$	ε	7 :	C'	$\to$	ε

An example of a derivation follows.

$$
\begin{array}{rlll}
S_I & \Rightarrow & ABC & [1] \\
& \Rightarrow & aABC & [2] \\
& \Rightarrow & aAbBC & [3] \\
& \Rightarrow & aAbBcC & [4] \\
& \Rightarrow & aaAbBcC & [2] \\
& \Rightarrow & aaAbbBcC & [3] \\
& \Rightarrow & aaAbbBccC & [4] \\
& \Rightarrow & aabbBccC & [5] \\
& \Rightarrow & aabbccC & [6] \\
& \Rightarrow & aabbcc & [7]
\end{array}
\qquad
\begin{array}{rlll}
S_O & \Rightarrow & A & [1] \\
& \Rightarrow & B & [2] \\
& \Rightarrow & C & [3] \\
& \Rightarrow & A & [4] \\
& \Rightarrow & B & [2] \\
& \Rightarrow & C & [3] \\
& \Rightarrow & A & [4] \\
& \Rightarrow & B' & [5] \\
& \Rightarrow & C' & [6] \\
& \Rightarrow & \varepsilon & [7]
\end{array}
$$

We can easily see that $L_I(H) = \{a^n b^n c^n : n \geq 0\}$, which is well known not to be a context-free language. This shows that RSCFGs are stronger than (synchronous) CFGs.[1] Where exactly do synchronous grammars with linked rules stand in terms of generative power?

Let $\mathscr{L}(\text{RSCFG})$ denote the class of languages generated by RSCFGs as their input language. Note that the results presented below would be the same if we considered the output language instead.

In some of the proofs below, we use a function that removes all terminals from a sentential form, formally defined as follows.

Definition 2.4. Let $G = (N, T, P, S)$ be a CFG. Then, we define the function θ over $(N \cup T)^*$ as follows:

1. For all $w \in T^*$, $\theta(w) = \varepsilon$.
2. For all $w = x_0 A_1 x_2 A_2 \ldots x_{n-1} A_n x_n$ for some $n \geq 1$, where $x_i \in T^*$ for all $0 \leq i \leq n$ and $A_j \in N$ for all $1 \leq j \leq n$, $\theta(w) = A_1 A_2 \ldots A_n$.

The idea here is that if we consider only context-free rules, the applicability of rules to a given sentential form only depends on nonterminals. Therefore, we can remove terminals without affecting computational control.

For every RSCFG, we can construct an equivalent MAT, using matrices to simulate the principle of linked rules.

Lemma 2.1. *For every RSCFG H, there is a MAT H' such that $L(H') = L_I(H)$.*

Proof. Let $H = (G_I, G_O, \Psi, \varphi_I, \varphi_O)$ be an RSCFG, where $G_I = (N_I, T_I, P_I, S_I)$, $G_O = (N_O, T_O, P_O, S_O)$. Without loss of generality, assume $N_I \cap N_O = \emptyset$, $S \notin N_I \cup N_O$. Construct a MAT $H' = (G, M)$, where $G = (N, T, P, S)$, as follows:

1. Set $N = N_I \cup N_O \cup \{S\}$, $T = T_I$, $P = \{S \to S_I S_O\}$, $M = \{S \to S_I S_O\}$.
2. For every label $p \in \Psi$, add rules p_I, p_O to P and add matrix $p_I p_O$ to M, where
   - $p_I = \varphi_I(p)$ and

[1] Strictly speaking, to make this claim, we also have to show that every context-free language can be generated by a RSCFG. That is however evident from the definition.

- $p_O = A \to x$ such that $\varphi_O(p) = A \to x'$, $x = \theta(x')$.[2]

Basic idea. H' simulates the principle of linked rules in H by matrices. That is, for every pair of rules $(A_I \to x_I, A_O \to x_O)$ such that $\varphi_I(p) = A_I \to x_I, \varphi_O(p) = A_O \to x_O$ for some $p \in \Psi$ in H, there is a matrix $m = A_I \to x_I A_O \to \theta(x_O)$ in H'. If, in H, $x_I \Rightarrow y_I\,[p]$ in G_I and $x_O \Rightarrow y_O\,[p]$ in G_O, then there is a derivation step $x_I\,\theta(x_O) \Rightarrow y_I\,\theta(y_O)\,[m]$ in H'. Note that since the rules are context-free, the presence (or absence) of terminals in a sentential form does not affect which rules we can apply. Furthermore, because the nonterminal sets N_I and N_O are disjoint, the sentential form in H' always consists of two distinct parts such that the first part corresponds to the derivation in G_I and the second part to the derivation in G_O.

The complete formal proof of $L(H') = L_I(H)$ can be found in the thesis. $\square$

On the other hand, for every MAT, we can construct an equivalent RSCFG. We take advantage of that fact that there is an "additonal" CFG in an RSCFG, and use it to simulate matrices.

Lemma 2.2. *For every MAT H, there is a RSCFG H' such that $L_I(H') = L(H)$.*

Proof. Let $H = (G, M)$ be a MAT, where $G = (N, T, P, S)$. Without loss of generality, assume $N \cap \{S_I, S_O, X\} = \emptyset$. Construct an RSCFG $H' = (G_I, G_O, \Psi, \varphi_I, \varphi_O)$, where $G_I = (N_I, T_I, P_I, S_I)$, $G_O = (N_O, T_O, P_O, S_O)$, as follows:

1. Set $N_I = N \cup \{S_I, X\}$, $T_I = T$, $P_I = \{S_I \to SX, X \to \varepsilon\}$, $N_O = \{S_O, X\}$, $T_O = \{\#\}$, $P_O = \{S_O \to X, X \to \#\}$, $\varphi_I = \emptyset$, $\varphi_O = \emptyset$.
2. Set $\Psi = \{0, 1\}$, $\varphi_I(0) = S_I \to SX$, $\varphi_O(0) = S_O \to X$, $\varphi_I(1) = X \to \varepsilon$, $\varphi_O(1) = X \to \#$.
3. For every matrix $m = p \in M$, where $p \in P$,
   (a) add rule p to P_I,
   (b) add rule $X \to X$ to P_O,
   (c) add new label $\langle m \rangle$ to Ψ, and
   (d) set $\varphi_I(\langle m \rangle) = p$, $\varphi_O(\langle m \rangle) = X \to X$.
4. For every matrix $m = p_1 \dots p_n \in M$, where $n > 1$ and $p_i \in P$ for all $1 \le i \le n$,
   (a) add rules $p_1, \dots, p_n$ to P_I,
   (b) add new nonterminals $\langle Xm \rangle_1, \dots, \langle Xm \rangle_{n-1}$ to N_O,
   (c) add rules $X \to \langle Xm \rangle_1$, $\langle Xm \rangle_1 \to \langle Xm \rangle_2$, $\dots$, $\langle Xm \rangle_{n-2} \to \langle Xm \rangle_{n-1}$, $\langle Xm \rangle_{n-1} \to X$ to P_O,
   (d) add new labels $\langle m \rangle_1, \dots, \langle m \rangle_n$ to Ψ, and
   (e) set φ_I and φ_O as follows:
      - $\varphi_I(\langle m \rangle_1) = p_1$, $\varphi_O(\langle m \rangle_1) = X \to \langle Xm \rangle_1$,
      - $\varphi_I(\langle m \rangle_i) = p_i$, $\varphi_O(\langle m \rangle_i) = \langle Xm \rangle_{i-1} \to \langle Xm \rangle_i$ for all $1 < i < n$, and

[2] This removes all terminals from the right-hand side of the rule. Note that if we leave the rule unchanged, we obtain the concatenation of the input and the output sentence. Further, if we want $L(H') = L_O(H)$ instead of $L(H') = L_I(H)$, we can simply modify p_I instead of p_O in this step.

- $\varphi_I(\langle m \rangle_n) = p_n$, $\varphi_O(\langle m \rangle_n) = \langle Xm \rangle_{n-1} \rightarrow X$.

Basic idea. One may notice that G_I constructed by the above algorithm is nearly identical to the original CFG G in H. Indeed, it performs essentially the same role: generating a sentence. Meanwhile, G_O restricts available derivations according to matrices from H. Each nonterminal in G_O represents a certain state of the system. For example, suppose that we have the nonterminal $\langle Xm \rangle_2$ as the current sentential form in G_O. This means that we are currently simulating the matrix m, we have succesfully applied the second rule of this matrix, and now we need to apply its next rule. The nonterminal X is a special case. It represents the state where we can either choose a new matrix to simulate, or end the derivation. It appears at the start of a derivation (along with the original start symbol from H, S) and can only appear again immediately after a successful simulation of a whole matrix (one derivation step in H).

In other words, H' simulates matrices in H by derivation in G_O. That is, if $x \Rightarrow y\,[m]$ in H, where $m = p_1 \dots p_n$ for some $n \geq 1$, then there is a sequence of derivation steps $X \Rightarrow \langle Xm \rangle_1\,[\langle m \rangle_1] \Rightarrow \langle Xm \rangle_2\,[\langle m \rangle_2] \Rightarrow \dots \Rightarrow \langle Xm \rangle_{n-2}\,[\langle m \rangle_{n-2}] \Rightarrow \langle Xm \rangle_{n-1}\,[\langle m \rangle_{n-1}] \Rightarrow X\,[\langle m \rangle_n]$ in G_O and $\varphi_I(\langle m \rangle_i) = p_i$ for $1 \leq i \leq n$. Now observe that in G_O constructed by the above algorithm, every nonterminal except X can only appear as the left-hand side of no more than one rule. This means that after rewriting X to $\langle Xm \rangle_1$, the only way for the derivation to proceed is the above sequence, and the entire matrix is simulated. Note that for matrices that only have one rule (that is, if $n = 1$), $X \Rightarrow X$ in G_O by using rule $X \rightarrow X$, and we can immediately continue with another matrix. The simulation ends by rewriting X to the only terminal $\#$ in G_O and, simultaneously, deleting X in G_I (using rule $X \rightarrow \varepsilon$). This ensures that the derivation in G_I cannot end by producing a sentence prematurely—that is, when the simulation of a matrix is incomplete—because there will always be at least one nonterminal left at that point (precisely X).

The complete formal proof of $L_I(H') = L(H)$ can be found in the thesis. □

Note that G_O constructed by the above algorithm is not only context-free, but also regular.

From Lemma 2.1 and Lemma 2.2, we can establish the following theorem.

Theorem 2.3.

$$\mathscr{L}(\text{RSCFG}) = \mathscr{L}(\text{MAT})$$

Proof. From Lemma 2.1, it follows that $\mathscr{L}(\text{RSCFG}) \subseteq \mathscr{L}(\text{MAT})$. From Lemma 2.2, it follows that $\mathscr{L}(\text{MAT}) \subseteq \mathscr{L}(\text{RSCFG})$. Therefore, $\mathscr{L}(\text{RSCFG}) = \mathscr{L}(\text{MAT})$. □

2.2 Synchronous Scattered Context Grammar

The principle of synchronization based on linked rules can be naturally extended to other models beside CFGs. Indeed, the definition of synchronous SCG is analogous to Definition 2.1 for RSCFG. Essentially, we only need to replace context-free rules with scattered context rules. The notation is also analogous.

Definition 2.5 (Synchronous SCG)**.** A *synchronous SCG* (SSCG for short) H is a quintuple $H = (G_I, G_O, \Psi, \varphi_I, \varphi_O)$, where

- $G_I = (N_I, T_I, P_I, S_I)$ and $G_O = (N_O, T_O, P_O, S_O)$ are SCGs,

- Ψ is a set of *rule labels*, and
- φ_I is a function from Ψ to P_I and φ_O is a function from Ψ to P_O.

Further, the *translation defined by* H, denoted by $T(H)$, is the set of pairs of sentences, which is defined as

$$\begin{aligned} T(H) = \{(w_I, w_O) \quad : \quad & w_I \in T_I^*, w_O \in T_O^*, \\ & S_I \Rightarrow_{G_I}^* w_I\,[\alpha], S_O \Rightarrow_{G_O}^* w_O\,[\alpha] \text{ for some } \alpha \in \Psi^*\}. \end{aligned}$$

We define the input and output language of SSCG by analogy with Definition 2.3 for RSCFGs. Further, let $\mathscr{L}(\mathrm{SSCG})$ denote the class of all languages generated by SSCGs as their input language.

2.2.1 Generative Power

It is known that SCGs can generate all recursively enumerable languages (see [29]). Perhaps not surprisingly, the same is true for SSCGs. From their definition, it is easy to see that SSCGs cannot be weaker than SCGs. If we want to construct an SSCG equivalent to a given SCG, we can, for instance, essentially duplicate the original SCG and designate each two identical rules from input and output grammar as linked.

Theorem 2.4.

$$\mathscr{L}(\mathrm{SSCG}) = \mathbf{RE}$$

Proof. Clearly, $\mathscr{L}(\mathrm{SSCG}) \subseteq \mathbf{RE}$ must hold. From definition, it follows that $\mathscr{L}(\mathrm{SCG}) \subseteq \mathscr{L}(\mathrm{SSCG})$. Because $\mathscr{L}(\mathrm{SCG}) = \mathbf{RE}$ [29], $\mathbf{RE} \subseteq \mathscr{L}(\mathrm{SSCG})$ also holds. □

2.3 Synchronous Matrix Grammar

In the case of matrix grammars, the situation is slightly more complicated. How should we link the rules with regard to matrices? There are many options. For instance, we could strictly require that all rules in one matrix in the input grammar be linked to rules in one matrix in the output grammar, in respective order (consequently, requiring each two matrices that have their rules linked to have the same length). Alternatively, we could link only the first rule in each matrix. However, perhaps the most straightforward and intuitive approach is to link whole matrices rather than individual rules.

The notation used here is analogous to the one presented in Section 2.1 for RSCFGs, only replacing rules by matrices.

Definition 2.6 (Synchronous matrix grammar)**.** A *synchronous matrix grammar* (SMAT for short) H is a septuple $H = (G_I, M_I, G_O, M_O, \Psi, \varphi_I, \varphi_O)$, where

- (G_I, M_I) and (G_O, M_O) are MATs, where
  - $G_I = (N_I, T_I, P_I, S_I)$ and
  - $G_O = (N_O, T_O, P_O, S_O)$,
- Ψ is a set of *matrix labels*, and
- φ_I is a function from Ψ to M_I and φ_O is a function from Ψ to M_O.

Further, the *translation defined by* H, denoted by $T(H)$, is the set of pairs of sentences, which is defined as

$$\begin{aligned} T(H) = \{(w_I, w_O) \quad : \quad & w_I \in T_I^*, w_O \in T_O^*, \\ & S_I \Rightarrow^*_{(G_I, M_I)} w_I\,[\alpha], S_O \Rightarrow^*_{(G_O, M_O)} w_O\,[\alpha] \text{ for some } \alpha \in \Psi^*\}. \end{aligned}$$

We define the input and output language of SMAT by analogy with Definition 2.3 for RSCFGs. Further, let $\mathscr{L}(\text{SMAT})$ denote the class of all languages generated by SMATs as their input language.

2.3.1 Generative Power

Following a similar reasoning as in the case of SSCGs, we can immediately conclude that SMATs must be at least as powerful as MATs. To elaborate, to construct an SMAT equivalent to a given MAT, we can, as with SSCGs, let both input and output grammar equal the original grammar and designate the identical matrices in input and output grammar as linked.

The fact that we can also construct an equivalent MAT for every SMAT is much less immediately obvious. In essence, we can join each two linked matrices (from input and output grammar) into one matrix.

Theorem 2.5.

$$\mathscr{L}(\text{SMAT}) = \mathscr{L}(\text{MAT})$$

Proof. The inclusion $\mathscr{L}(\text{MAT}) \subseteq \mathscr{L}(\text{SMAT})$ follows from definition. It only remains to prove that $\mathscr{L}(\text{SMAT}) \subseteq \mathscr{L}(\text{MAT})$. For every SMAT $H = (G_I, M_I, G_O, M_O, \Psi, \varphi_I, \varphi_O)$, where $G_I = (N_I, T_I, P_I, S_I)$, $G_O = (N_O, T_O, P_O, S_O)$, we can construct a MAT $H' = (G, M)$, where $G = (N, T, P, S)$, such that $L(H') = L_I(H)$, as follows. Without loss of generality, assume $N_I \cap N_O = \emptyset$, $S \notin N_I \cup N_O$.

1. Set $N = N_I \cup N_O \cup \{S\}$, $T = T_I$, $P = \{S \to S_I S_O\}$, $M = \{S \to S_I S_O\}$.
2. For every label $p \in \Psi$, add rules $p_{I1}, \dots, p_{In}$, $p_{O1}, \dots, p_{Om}$ to P and add matrix $p_{I1} \dots p_{In} p_{O1} \dots p_{Om}$ to M, where
   - $p_{I1} \dots p_{In} = \varphi_I(p)$ and
   - for $1 \leq j \leq m$, $p_{Oj} = A_j \to x_j$ such that $\varphi_O(p)[j] = A_j \to x'_j$, $x_j = \theta(x'_j)$.[3]

Basic idea. H' simulates H by combining the rules of each two linked matrices in H into a single matrix in H'. That is, for every pair of matrices (m_I, m_O) such that $m_I = \varphi_I(p), m_O = \varphi_O(p)$ for some $p \in \Psi$ in H, there is a matrix $m = m_I m'_O$ in H', where m'_O is equal to m_O with all terminals removed (formally defined above). If, in H, $x_I \Rightarrow y_I\,[p]$ in G_I and $x_O \Rightarrow y_O\,[p]$ in G_O, then there is a derivation step $x_I\,\theta(x_O) \Rightarrow y_I\,\theta(y_O)\,[m]$ in H'. Note that since the rules are context-free, the presence (or absence) of terminals in a sentential form does not affect which rules we can apply. Furthermore, because the nonterminal sets N_I and N_O are disjoint, the sentential form in H' always consists of two distinct parts such that the first part corresponds to the derivation in G_I and the second part to the derivation in G_O.

The complete formal proof of $L(H') = L_I(H)$ can be found in the thesis. □

[3]Again, this removes all terminals from the right-hand side of the rules (see Theorem 2.3). $m[j]$ denotes the j-th rule in matrix m.

Chapter 3

Synchronous Systems Based on Transducers

In formal language theory, there exist two basic translation-method categories. The first category contains interprets and compilers, which first analyse an input string in the source language and, consequently, they generate a corresponding output string in the target language (see [2], [25], [34], [37], or [40]). The second category is composed of language-translation systems or, more briefly, transducers. Frequently, these trasducers consist of several components, including various automata and grammars, some of which read their input strings while others produce their output strings (see [15], [36], and [41]).

Although transducers represent language-translation devices, language theory often views them as language-defining devices and investigates the language family resulting from them. That is, it studies their accepting power consisting in determining the language families accepted by the transducer components that read their input strings. Alternatively, it establishes their generative power that determines the language family generated by the components that produce their strings. The present chapter contributes to this vivid investigation trend in formal language theory.

In this chapter, we introduce a new type of transducer, referred to as rule-restricted (automaton-grammar) transducer, based upon an FA and a CFG. We discuss the power of this system working in an ordinary way as well as in a leftmost way and investigate an effect of an appearance checking placed into the system.

The original ideas, concepts, definitions, and theoretical results presented in this chapter were first published in [6].

3.1 Rule-Restricted Transducer

The rule-restricted (automaton-grammar) transducer is a hybrid system consisting based on a straightforward idea: we read an input sentence with an FA while generating an appropriate output sentence with a CFG. A control set determines which rules from the FA and the CFG can be used simultaneously. The computation of the system is successful if and only if the FA accepts the input string and the CFG generates a string of terminals.

Definition 3.1 (Rule-restricted transducer)**.** The *rule-restricted transducer* (RT for short) Γ is a triple $\Gamma = (M, G, \Psi)$, where

- $M = (Q, \Sigma, \delta, q_0, F)$ is an FA,

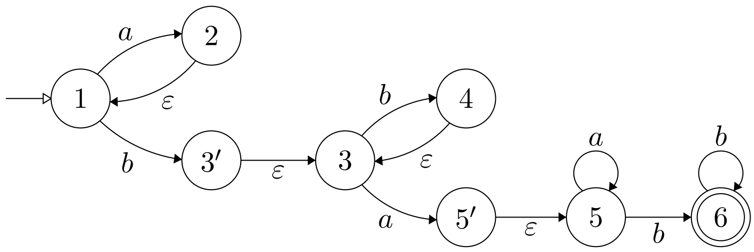

Figure 3.1: Definition of FA M from Example 3.1

- $G = (N, T, P, S)$ is a CFG, and
- Ψ is a finite set of pairs of the form (r_1, r_2), where r_1 and r_2 are rules from δ and P, respectively.

A 2-*configuration* of Γ is a pair $\chi = (x, y)$, where $x \in Q\Sigma^*$ and $y \in (N \cup T)^*$. Consider two 2-configurations, $\chi = (pav_1, uAv_2)$ and $\chi' = (qv_1, uxv_2)$ with $A \in N$, $u, v_2, x \in (N \cup T)^*$, $v_1 \in \Sigma^*$, $a \in \Sigma \cup \{\varepsilon\}$, and $p, q \in Q$. If $pav_1 \Rightarrow qv_1\, [r_1]$ in M, $uAv_2 \Rightarrow uxv_2\, [r_2]$ in G, and $(r_1, r_2) \in \Psi$, then Γ makes a computation step from χ' to χ', written as $\chi \Rightarrow \chi'$. In the standard way, $\Rightarrow^*$ and $\Rightarrow^+$ are transitive-reflexive and transitive closure of $\Rightarrow$, respectively.

The 2-*language of* Γ, 2-$L(\Gamma)$, is 2-$L(\Gamma) = \{(w_1, w_2)\colon\ (q_0 w_1, S) \Rightarrow^* (f, w_2),\ w_1 \in \Sigma^*,\ w_2 \in T^*$, and $f \in F\}$. From the 2-language we can define two languages:

- $L(\Gamma)_1 = \{w_1 \colon (w_1, w_2) \in$ 2-$L(\Gamma)\}$, and
- $L(\Gamma)_2 = \{w_2 \colon (w_1, w_2) \in$ 2-$L(\Gamma)\}$.

By $\mathscr{L}(\mathrm{RT})$, $\mathscr{L}(\mathrm{RT})_1$, and $\mathscr{L}(\mathrm{RT})_2$, the classes of 2-languages of RTs, languages accepted by M in RTs, and languages generated by G in RTs, respectively, are understood.

3.1.1 Generative Power

It is well-known that FAs and CFGs describe different classes of languages. Specifically, by FAs we can accept regular languages, whereas CFGs define the class of context-free languages. However, in Example 3.1 it is shown that by the combination of these two models, the system is able to accept and generate even non-context-free languages.

Example 3.1. Consider RT $K = (M, G, \Psi)$ with

- M given by graphical representation in Figure 3.1
- $G = (\{S, A, B, C, D, D'\}, \{a, b\}, P, S)$, where

  – $$P = \left\{ \begin{array}{lll} r_1 : S \to BbD', & r_2 : B \to Bb, & r_3 : D' \to D'D, \\ r_4 : B \to aA, & r_5 : D' \to C, & r_6 : A \to aA, \\ r_7 : C \to CC, & r_8 : D \to b, & r_9 : A \to \varepsilon, \\ r_{10} : C \to a & & \end{array} \right\}$$
- $\Psi = \{(p_1, r_1),\ (p_1, r_2),\ (p_2, r_3),\ (p_3, r_4),\ (p_4, r_5),\ (p_5, r_6),\ (p_6, r_7),\ (p_7, r_8),\ (p_8, r_9),\ (p_9, r_8),\ (p_{10}, r_{10}),\ (p_{11}, r_{10})\}$.

The languages of M and G are $L(M) = \{a^i b^j a^k b^l \colon\ j, k, l \in \mathbb{N}, i \in \mathbb{N}_0\}$ and $L(G) = \{a^i b^j a^k b^l \colon\ i, j, k \in \mathbb{N}, l \in \mathbb{N}_0\}$, respectively. However, the 2-language of K is $L(K) = \{(a^i b^j a^i b^j, a^j b^i a^j b^i) \colon\ i, j \in \mathbb{N}\}$.

From the example, observe that the power of the grammar increases due to the possibility of synchronization with the automaton that can dictate sequences of usable rules in the grammar. The synchronization with the automaton enhances the generative power of the grammar up to the class of languages generated by MATs.

Theorem 3.1.

$$\mathscr{L}(\mathrm{RT})_2 = \mathscr{L}(\mathrm{MAT})$$

Proof. I. First we prove that $\mathscr{L}(\mathrm{MAT}) \subseteq \mathscr{L}(\mathrm{RT})_2$.

Consider a MAT $I = ({}_IG, {}_IC)$ and construct an RT $\Gamma = ({}_\Gamma M, {}_\Gamma G, \Psi)$, such that $L(I) = L(\Gamma)_2$, as follows. Set ${}_\Gamma G = {}_I G$. Construct ${}_\Gamma M = (Q, \Sigma, \delta, s, F)$ in the following way:

1. Set $F, Q = \{s\}$.
2. For every $m = p_1 \dots p_k \in {}_IC$, add:
   (a) $k - 1$ new states, $q_1, q_2, \dots, q_{k-1}$, into Q,
   (b) k new rules, $r_1 = s \to q_1, r_2 = q_1 \to q_2, \dots, r_{k-1} = q_{k-2} \to q_{k-1}, r_k = q_{k-1} \to s$, into δ, and
   (c) k new pairs, $(r_1, p_1), (r_2, p_2), \dots, (r_{k-1}, p_{k-1})$, (r_k, p_k), into Ψ.

The FA ${}_\Gamma M$ simulates matrices in I by transitions. That is, if $x_1 \Rightarrow x_2\,[p]$ in I, where $p = p_1, \dots, p_i$ for some $i \in \mathbb{N}$, then there is $q_1, \dots, q_{i-1} \in Q$ such that $r_1 = s \to q_1, r_2 = q_1 \to q_2, \dots, r_{i-1} = q_{i-2} \to q_{i-1}, r_i = q_{i-1} \to s \in \delta$ and $(r_1, p_1), \dots, (r_i, p_i) \in \Psi$. Therefore, $(s, x_1) \Rightarrow^i (s, x_2)$ in Γ. Similarly, if $(s, x_1) \Rightarrow^i (s, x_2)$ in Γ, for $i \in \mathbb{N}$, and there is no $j \in \mathbb{N}$ such that $0 < j < i$ and $(s, x_1) \Rightarrow^j (s, y) \Rightarrow^* (s, x_2)$, there has to be $p \in {}_IC$ and $x_1 \Rightarrow x_2\,[p]$ in I. Hence, if $(s, S) \Rightarrow^* (s, w)$ in Γ, where w is a string over the set of terminals in ${}_\Gamma G$, then $S \Rightarrow^* w$ in I; and, on the other hand, if $S \Rightarrow^* w$ in I for a string over the set of terminals in ${}_IG$, then $(s, S) \Rightarrow^* (s, w)$ in Γ. The inclusion $\mathscr{L}(\mathrm{MAT}) \subseteq \mathscr{L}(\mathrm{RT})_2$ has been proven.

II. Next, we prove the inclusion $\mathscr{L}(\mathrm{RT})_2 \subseteq \mathscr{L}(\mathrm{MAT})$. For any RT $\Gamma = ({}_\Gamma M = (Q, \Sigma, \delta, s, F), {}_\Gamma G = ({}_\Gamma N, {}_\Gamma T, {}_\Gamma P, {}_\Gamma S), \Psi)$, we can construct a MAT $O = ({}_OG, {}_OC)$ such that $L(\Gamma)_2 = L(O)$ as follows:

1. Set ${}_OG = ({}_\Gamma N \cup \{S'\}, {}_\Gamma T, {}_OP, S')$, ${}_OP = {}_\Gamma P \cup \{p_0 = S' \to \langle s \rangle {}_\Gamma S\}$, and ${}_OC = \{p_0\}$.
2. For each pair $(p_1, p_2) \in \Psi$ with $p_1 = qa \to r$, $q, r \in Q$, $a \in \Sigma \cup \{\varepsilon\}$, $p_2 = A \to x$, $A \in {}_\Gamma N$ and $x \in ({}_\Gamma N \cup {}_\Gamma T)^*$, add $p_1 = \langle q \rangle \to \langle r \rangle$ into ${}_OP$ and $p_1 p_2$ into ${}_OC$.
3. Furthermore, for all $q \in F$, add $p = \langle q \rangle \to \varepsilon$ into ${}_OP$ and p into ${}_OC$.

The complete formal proof of $L(\Gamma)_2 = L(H)$ can be found in the thesis. □

3.1.2 Accepting Power

On the other hand, the CFG in the RT can be exploited as an additional storage space of the FA to remember some non-negative integers. If the automaton uses the CFG in this way, the additional storage space is akin to counters in a multi-counter machine. The following lemma says that the FAs in RTs are able to accept every language accepted by partially blind k-counter automata.

Lemma 3.2. *For every k-PBCA I, there is an RT $\Gamma = (M, G, \Psi)$ such that $L(I) = L(\Gamma)_1$.*

Proof of Lemma 3.2. Let $I = ({}_IQ, \Sigma, {}_I\delta, q_0, F)$ be a k-PBCA for some $k \geq 1$ and construct a RT $\Gamma = (M = ({}_MQ, \Sigma, {}_M\delta, q_0, F), G = (N, T, P, S), \Psi)$ as follows:

1. Set $T = \emptyset$, $\Psi = \emptyset$, $N = \{S, A_1, \dots, A_k\}$, $P = \{A \to \varepsilon \colon\ A \in N\}$, ${}_M\delta = \{f \to f \colon\ f \in F\}$, and ${}_MQ = {}_IQ$.
2. For each $pa \to q(t_1, \dots, t_k)$ in ${}_I\delta$ and for $n = (\Sigma_{i=1}^{k} \max(0, -t_i))$ add:
   (a) $q_1, \dots, q_n$ into ${}_MQ$;
   (b) $r = S \to xS$, where $x \in (N - \{S\})^*$ and $\mathrm{occur}(A_i, x) = \max(0, t_i)$, for $i = 1, \dots, k$, into P;
   (c) $r_1 = q_0 a \to q_1$, $r_2 = q_1 \to q_2$, …, $r_n = q_{n-1} \to q_n$, $r_{n+1} = q_n \to q$ into ${}_M\delta$ with $q_0 = p$; and $(r_{i+1}, \alpha_i \to \varepsilon)$, where $\alpha_i = A_j$ and each A_j is erased $\max(0, -t_i)$-times during the sequence, into Ψ ($n = 0$ means that only $pa \to q$, $S \to xS$ and (r_1, r) are considered);
   (d) $(f \to f, S \to \varepsilon)$ into Ψ for all $f \in F$.

The FA of the created system uses the CFG as an external storage. Each counter of I is represented by a nonterminal. Every step from p to q that modifies counters is simulated by several steps leading from p to q and during this sequence of steps the number of occurrences of each nonterminal in the grammar is modified to be equal to the corresponding counter in I. Clearly, $L(I) = L(\Gamma)_1$. □

Lemma 3.3 states that the CFG is helpful for the FA in RT at most with the preservation of the non-negative numbers without possibility to check their values.

Lemma 3.3. *For every RT $\Gamma = (M, G, \Psi)$, there is a k-PBCA O such that $L(O) = L(\Gamma)_1$ and k is the number of nonterminals in G.*

Proof of Lemma 3.3. Let $\Gamma = (M = (Q, \Sigma, {}_M\delta, q_0, F), G = (N, T, P, S), \Psi)$ be an RT. Without any loss of generality, suppose that $N = \{A_1, \dots, A_n\}$, where $S = A_1$. The partially blind $\mathrm{card}(N)$-counter automaton $O = (Q, \Sigma, {}_O\delta, q_0, F)$ is created in the following way. For each $r_1 = pa \to q \in {}_M\delta$ and $r_2 = \alpha \to \beta \in P$ such that $(r_1, r_2) \in \Psi$, add $pa \to q(v_1, \dots, v_{\mathrm{card}(N)})$, where $v_i = \mathrm{occur}(A_i, \beta) - \mathrm{occur}(A_i, \alpha)$ for all $i = 1, \dots, \mathrm{card}(N)$.

The constructed partially blind $\mathrm{card}(N)$-counter automaton has a counter for each nonterminal from the grammar of Γ. Whenever the automaton in Γ makes a step and thes entential form of the grammar G is changed, O makes the same step and accordingly changes the number of occurrences of nonterminals in its counters. □

From Lemma 3.2 and Lemma 3.3, we can establish the following theorem.

Theorem 3.4.

$$\mathscr{L}(\mathrm{RT})_1 = \bigcup_{k=1}^{\infty} \mathscr{L}(k\text{-PBCA})$$

Proof. It directly follows from Lemma 3.2 and Lemma 3.3. □

For better illustration of the accepting and generative power of RT, let us recall that the class of languages generated by MATs is properly included in the class of **RE** languages [1, 12], and the class of languages defined by partially blind k-counter automata, with respect to number of counters, is superset of the class of **CF** languages and properly included in the class of **CS** languages [13, 14].

3.2 Rule-Restricted Transducer with Leftmost Restriction

Although the investigated system is relatively powerful, in defiance of weakness of models that are used, nondeterministic selections of nonterminals to be rewritten can be relatively problematic from the practical point of view. Therefore, we examine an effect of a restriction in the form of leftmost derivations placed on the CFG in RT.

Definition 3.2 (Leftmost restriction on derivation in RT)**.** Let $\Gamma = (M, G, \Psi)$ be an RT with $M = (Q, \Sigma, \delta, q_0, F)$ and $G = (N, T, P, S)$. Furthermore, let $\chi = (pav_1, uAv_2)$ and $\chi' = (qv_1, uxv_2)$ be two 2-configurations, where $A \in N$, $v_2, x \in (N \cup T)^*$, $u \in T^*$, $v_1 \in \Sigma^*$, $a \in \Sigma \cup \{\varepsilon\}$, and $p, q \in Q$. Γ makes a computation step from χ to χ', written as $\chi \Rightarrow_{lm} \chi'$, if and only if $pav_1 \Rightarrow qv_1\, [r_1]$ in M, $uAv_2 \Rightarrow uxv_2\, [r_2]$ in G, and $(r_1, r_2) \in \Psi$. In the standard way, $\Rightarrow_{lm}^*$ and $\Rightarrow_{lm}^+$ are transitive-reflexive and transitive closure of $\Rightarrow_{lm}$, respectively.

The 2-*language of* Γ with G generating in the leftmost way, denoted by 2-$L_{lm}(\Gamma)$, is defined as 2-$L_{lm}(\Gamma) = \{(w_1, w_2)\colon\ (q_0w_1, S) \Rightarrow_{lm}^* (f, w_2),\ w_1 \in \Sigma^*,\ w_2 \in T^*, \text{ and } f \in F\}$; we call Γ a *leftmost restricted RT*; and we define the languages given from 2-$L_{lm}(\Gamma)$ as $L_{lm}(\Gamma)_1 = \{w_1\colon\ (w_1, w_2) \in 2\text{-}L_{lm}(\Gamma)\}$ and $L_{lm}(\Gamma)_2 = \{w_2\colon\ (w_1, w_2) \in 2\text{-}L_{lm}(\Gamma)\}$.

By $\mathscr{L}(\mathrm{RT}_{lm})$, $\mathscr{L}(\mathrm{RT}_{lm})_1$, and $\mathscr{L}(\mathrm{RT}_{lm})_2$, we understand the following language classes, respectively: 2-languages of leftmost restricted RTs, languages accepted by M in leftmost restricted RTs, and languages generated by G in leftmost restricted RTs.

3.2.1 Generative Power

Unfortunately, the price for the leftmost restriction, placed on derivations in the CFG, is relatively high and both accepting and generative ability of RT with the restriction decreases to the definition of context-free languages.

Theorem 3.5.

$$\mathscr{L}(\mathrm{RT}_{lm})_2 = \mathbf{CF}$$

Proof. The inclusion $\mathbf{CF} \subseteq \mathscr{L}(\mathrm{RT}_{lm})_2$ is clear from the definition, because any time we can construct leftmost restricted RT, where the automaton M cycles with reading all possible symbols from the input or ε whilst the grammar G is generating some output string. Therefore, we only need to prove the opposite inclusion.

We know that the class of context-free languages is defined, inter alia, by nondeterministic PDAs. It is therefore sufficient to prove that every language $L_{lm}(\Gamma)_2$ of RT can be accepted by a nondeterministic PDA. Consider an RT $\Gamma = ({}_\Gamma M = (Q, {}_\Gamma\Sigma, {}_\Gamma\delta, q_0, F), G = (N, T, P, S), \Psi)$ and define a PDA $O = (Q, T, {}_O\Gamma, {}_O\delta, q_0, S, F)$, where ${}_O\Gamma = N \cup T$ and ${}_O\delta$ is created as follows:

1. Set ${}_O\delta = \emptyset$.
2. For each $r_1 = A \to x \in P$ and $r_2 = pa \to q \in {}_\Gamma\delta$ such that $(r_1, r_2) \in \Psi$, add $Ap \to (x)^R q$ into ${}_O\delta$.
3. For each $p \in Q$, and $a \in T$ add $apa \to p$ into ${}_O\delta$.

The complete formal proof of $L(O) = L_{lm}(\Gamma)_2$ can be found in the thesis.
As $L(O) \subseteq L_{lm}(\Gamma)_2$ and $L_{lm}(\Gamma)_2 \subseteq L(O)$, Theorem 3.5 holds. □

3.2.2 Accepting Power

First, we show that any context-free language can be accepted by some leftmost restricted RT.

Lemma 3.6. *For every language $L \in \mathbf{CF}$, there is an RT $\Gamma = (M, G, \Psi)$ such that $L_{lm}(\Gamma)_1 = L$.*

Proof of Lemma 3.6. Let $I = ({_I N}, T, {_I P}, S)$ be a CFG such that $L(I) = L$. For I, we can construct a CFG $H = ({_H N}, T, {_H P}, S)$, where ${_H N} = {_I N} \cup \{\langle a \rangle \colon\ a \in T\}$ and ${_H P} = \{\langle a \rangle \to a \colon\ a \in T\} \cup \{A \to x \colon\ A \to x' \in {_I P}$ and x is created from x' by replacing all $a \in T$ in x' with $\langle a \rangle\}$. Surely, $L(I) = L(H)$ even if H replaces only the leftmost nonterminals in each derivation step. In addition, we construct an FA $M = (\{q_0\}, T, \delta, q_0, \{q_0\})$ with $\delta = \{q_0 \to q_0\} \cup \{q_0 a \to q_0 \colon\ a \in T\}$, and $\Psi = \{(q_0 \to q_0, A \to x) \colon\ A \to x \in {_H P}, A \in {_I N}\} \cup \{(q_0 a \to q_0, \langle a \rangle \to a) \colon\ a \in T\}$.

It is easy to see that any time when H replaces nonterminals from ${_I N}$ in its sentential form, M reads no input symbol. If and only if H replaces $\langle a \rangle$ with a, where $a \in T$, then M reads a from the input. Since H works in a leftmost way, 2-$L_{lm}(\Gamma) = \{(w, w) \colon\ w \in L(I)$. Hence, $L_{lm}(\Gamma)_1 = L(I)$. □

Similarly, we show that any RT generating outputs in the leftmost way can recognize no language out of **CF**.

Lemma 3.7. *Let Γ is an RT. Then, for every language $L_{lm}(\Gamma)_1$, there is a PDA O such that $L_{lm}(\Gamma)_1 = L(O)$.*

Proof of Lemma 3.7. In the same way as in the proof of Theorem 3.1, we construct PDA O such that $L(O) = L_{lm}(\Gamma)_1$ for RT $\Gamma = (M = (Q, {_\Gamma \Sigma}, {_\Gamma \delta}, q_0, F), G = (N, T, P, S), \Psi)$. We define O as $O = (Q, {_\Gamma \Sigma}, N, {_O \delta}, q_0, S, F)$, where ${_O \delta}$ is created in the following way:

1. Set ${_O \delta} = \emptyset$.
2. For each $r_1 = pa \to q \in {_\Gamma \delta}$ and $r_2 = A \to x \in P$ such that $(r_1, r_2) \in \Psi$, add $Apa \to (\theta(x))^R q$ into ${_O \delta}$, where $\theta(x)$ is a function from $(N \cup T)^*$ to N^* that replaces all terminal symbols in x with ε—that is, $\theta(x)$ is x without terminal symbols.[1]

The complete formal proof of $L(O) = L_{lm}(\Gamma)_2$ can be found in the thesis. □

Theorem 3.8.

$$\mathscr{L}(\mathrm{RT}_{lm})_1 = \mathbf{CF}$$

Proof. It directly follows from Lemma 3.6 and Lemma 3.7. □

3.3 Rule-Restricted Transducer with Appearance Checking

We can also extend RT with the possibility to prefer a rule over another—that is, the restriction sets contain triples of rules (instead of pairs of rules), where the first rule is a rule of FA, the second rule is a main rule of CFG, and the third rule is an alternative rule of CFG, which is used only if the main rule is not applicable.

[1]See page 10 for further explanation and precise formal definition of θ (Definition 2.4).

Definition 3.3 (RT with appearance checking)**.** *RT with appearance checking* (RT_{ac} for short) Γ is a triple $\Gamma = (M, G, \Psi)$, where

- $M = (Q, \Sigma, \delta, q_0, F)$ is an FA,
- $G = (N, T, P, S)$ is a CFG, and
- Ψ is a finite set of triples of the form (r_1, r_2, r_3) such that $r_1 \in \delta$ and $r_2, r_3 \in P$.

Let $\chi = (pav_1, uAv_2)$ and $\chi' = (qv_1, uxv_2)$, where $A \in N$, $v_2, x, u \in (N \cup T)^*$, $v_1 \in \Sigma^*$, $a \in \Sigma \cup \{\varepsilon\}$, and $p, q \in Q$, be two 2-configurations. Γ makes a computation step from χ to χ', written as $\chi \Rightarrow \chi'$, if and only if for some $(r_1, r_2, r_3) \in \Psi$, $pav_1 \Rightarrow qv_1\,[r_1]$ in M, and either

- $uAv_2 \Rightarrow uxv_2\,[r_2]$ in G, or
- $uAv_3 \Rightarrow uxv_2\,[r_3]$ in G and r_2 is not applicable on uAv_2 in G.

The 2-language 2-$L(\Gamma)$ and languages $L(\Gamma)_1, L(\Gamma)_2$ are defined in the same way as in Definition 3.1. The classes of languages defined by the first and the second component in the system is denoted by $\mathscr{L}(\mathrm{RT}_{ac})_1$ and $\mathscr{L}(\mathrm{RT}_{ac})_2$, respectively.

3.3.1 Generative Power

By the appearance checking both generative and accepting power of RT grow to define the class of all recursively enumerable languages. To prove that the former holds, we take advantage of the known fact that matrix grammars with appearance checking can generate any language in **RE** [12], and show that, in turn, RT_{ac} can simulate MAT_{ac}.

Theorem 3.9.

$$\mathscr{L}(\mathrm{RT}_{ac})_2 = \mathbf{RE}$$

Proof. Since $\mathscr{L}(\mathrm{MAT}_{ac}) = \mathbf{RE}$ [12], we only need to prove that $\mathscr{L}(\mathrm{MAT}_{ac}) \subseteq \mathscr{L}(\mathrm{RT}_{ac})_2$.

Consider a MAT_{ac} with appearance checking $I = ({}_IG, {}_IC)$ and construct a RT $\Gamma = ({}_\Gamma M, {}_\Gamma G, \Psi)$, such that $L(I) = L(\Gamma)_2$, as follows:

1. Set ${}_\Gamma G = {}_IG$.
2. Add a new initial nonterminal S', nonterminal Δ, and rules $\Delta \to \Delta$, $\Delta \to \varepsilon$, $S' \to S\Delta$ into grammar ${}_\Gamma G$.
3. Construct an FA ${}_\Gamma M = (Q, \Sigma, \delta, s, F)$ and Ψ in the following way:
   (a) Set $F = Q = \{s\}$, $\delta = \{s \to s\}$, and $\Psi = \{(s \to s, \Delta \to \varepsilon, \Delta \to \varepsilon), (s \to s, S' \to S\Delta, S' \to S\Delta)\}$.
   (b) For every $m = (p_1, t_1) \dots (p_k, t_k) \in {}_IC$, add $q_1, q_2, \dots, q_{k-1}$ into Q, $s \to q_1, q_1 \to q_2, \dots, q_{k-2} \to q_{k-1}, q_{k-1} \to s$ into δ, and $(s \to q_1, p_1, c_1), (q_1 \to q_2, p_2, c_2), \dots, (q_{k-2} \to q_{k-1}, p_{k-1}, c_{k-1})$, $(q_{k-1} \to q_s, p_k, c_k)$ into Ψ, where, for $1 \leq i \leq k$, if $t_i = -$, then $c_i = p_i$; otherwise, $c_i = \Delta \to \Delta$.

Since S' is the initial symbol, the first computation step in Γ is $(s, S') \Rightarrow (s, S\Delta)$. After this step, the FA simulates matrices in I by computation step. That is, if $x_1 \Rightarrow x_2\,[p]$ in I, where $p = p_1, \dots, p_i$ for some $i \in \mathbb{N}$, then there is $q_1, \dots, q_{i-1} \in Q$ such that $r_1 = s \rightarrow q_1, r_2 = q_1 \rightarrow q_2, \dots, r_{i-1} = q_{i-2} \rightarrow q_{i-1}, r_i = q_{i-1} \rightarrow s \in \delta$ and $(r_1, p_1, c_1), \dots, (r_i, p_i, c_i) \in \Psi$. Therefore, $(s, x_1) \Rightarrow^i (s, x_2)$ in Γ. Notice that if I can overleap some grammar rule in $m \in {}_IC$, Γ represents the fact by using $\Delta \rightarrow \Delta$ with the move in ${}_\Gamma M$. Similarly, if, for some $i \in \mathbb{N}$, $(s, x_1) \Rightarrow^i (s, x_2)$ in Γ and there is no $j < i$ such that $(s, x_1) \Rightarrow^j (s, y) \Rightarrow^* (s, x_2)$, there exists $p \in {}_IC$ such that $x_1 \Rightarrow x_2\,[p]$ in I. Hence, if $(s, S) \Rightarrow^* (s, w)$ in Γ, where w is a string over the set of terminals in ${}_\Gamma G$, then $S \Rightarrow^* w$ in I; and, on the other hand, if $S \Rightarrow^* w$ in I for a string over the set of terminals in ${}_IG$, then $(s, S') \Rightarrow (s, S\Delta) \Rightarrow^* (s, w\Delta) \Rightarrow (s, w)$ in Γ. $\square$

3.3.2 Accepting Power

RT$_{ac}$'s can accept any recursively enumerable language, as evidenced by their ability to simulate k-CAs.

Theorem 3.10.

$$\mathscr{L}(\mathrm{RT}_{ac})_1 = \mathbf{RE}$$

Proof. Let $I = ({}_IQ, \Sigma, {}_I\delta, q_0, F)$ be a k-CA for some $k \geq 1$ and construct a RT $\Gamma = (M, G, \Psi)$, where $M = ({}_MQ, \Sigma, {}_M\delta, q_0, F)$, $G = (N, T, P, S)$, as follows:

1. Set $T = \{a\}, \Psi = \emptyset, P = \{A \rightarrow \varepsilon, A \rightarrow \Diamond\colon\ A \in N - \{\Diamond\}\} \cup \{S \rightarrow S\}$, ${}_MQ = {}_IQ$, ${}_M\delta = \{f \rightarrow f\colon\ f \in F\}$, and $N = \{S, \Diamond, A_1, \dots, A_k\}$.
2. For each $pa \rightarrow q(t_1, \dots, t_k)$ in ${}_I\delta$, $n = \Sigma_{i=1}^k \theta(t_i)$, and $m = \Sigma_{i=1}^k \widehat{\theta}(t_i)$, where if $t_i \in \mathbb{Z}$, $\theta(t_i) = \max(0, -t_i)$ and $\widehat{\theta}(t_i) = \max(0, t_i)$; otherwise $\theta(t_i) = 1$ and $\widehat{\theta}(t_i) = 0$, add:
   (a) $q_1, \dots, q_n$ into ${}_MQ$;
   (b) $r = S \rightarrow xS$, where $x \in (N - \{S, \Diamond\})^*$ and $\mathrm{occur}(A_i, x) = \widehat{\theta}(t_i)$, for each $i = 1, \dots, k$, into P;
   (c) $r_1 = q_0a \rightarrow q_1$, $r_2 = q_1 \rightarrow q_2$, $\dots$, $r_n = q_{n-1} \rightarrow q_n$, $r_{n+1} = q_n \rightarrow q$ into ${}_M\delta$ with $q_0 = p$; and for each $i = 1, \dots, n$, add $(r_{i+1}, \tau_i, \tau_i')$, where for each $j = 1, \dots, k$, if $t_j \in \mathbb{N}$, for $\theta(t_j)$ is, $\tau_i = \tau_i' = A_j \rightarrow \varepsilon$; otherwise, if $t_j = -$, $\tau_i = A_j \rightarrow \Diamond$ and $\tau_i' = S \rightarrow S$, into Ψ. Notice that $n = 0$ means that only $q_0a \rightarrow q$, $S \rightarrow xS$ are considered. Furthermore, add (r_1, r, r) into Ψ;
   (d) $(f \rightarrow f, S \rightarrow \varepsilon, S \rightarrow \varepsilon)$ into Ψ for all $f \in F$.

Similarly as in the proof of Lemma 3.2, the FA of the created system uses the CFG as an external storage, and each counter of I is represented by a nonterminal. If I modifies some counters during a move from state p to state q, M moves from p to q in several steps during which it changes the numbers of occurrences of nonterminals correspondingly. Rules applicable only if some counters are equal to zero are simulated by using an appearance checking, where Γ tries to replace all nonterminals representing counters which have to be 0 by $\Diamond$. If it is not possible, Γ applies the rule $S \rightarrow S$ and continues with computation. Otherwise, since $\Diamond$ cannot be rewritten during the rest of computation, the use of such rules leads to an unsuccessful computation. The formal proof of the equivalence of languages is left to the reader. Since $\mathscr{L}(k\text{-CA}) = \mathbf{RE}$ for every $k \geq 2$ [17], Theorem 3.10 holds. $\square$

Chapter 4

Linguistic Applications: Perspectives

In this chapter, we discuss the advantages of the new formal models in regard to their potential applications in natural language processing, and particularly in translation, where they can provide an alternative to the existing models (see Chapter 3 of the full thesis for an overview). To illustrate, we use examples from Czech, English, and Japanese. (No prior knowledge of Czech or Japanese is required for understanding, although it can be an advantage.)

Throughout the course of this chapter, we use the following notation to represent some common linguistic constituents:

ADJ	adjective	ADV	adverb
AUX	auxiliary verb	DET	determiner
N	noun	NP	noun phrase
NP-SBJ	NP in the role of subject	NUM	numeral
P	preposition	PN	pronoun
PN-INT	interrogative pronoun	PP	prepositional phrase
PP-TMP	PP, temporal	PP-DIR	PP, directional
V	verb	VP	verb phrase

Further, note that in the example sentences presented below, we generally disregard punctuation and capitalization. For example, we consider *Where are you going?* and *where are you going* identical for the purposes of this text.

Finally, in most of the case studies presented in this chapter, we assume that we already have the input sentence split into words (or possibly some other lexical units as appropriate), and these words are classified as, for example, a noun, pronoun, or verb. Then, we consider syntax analysis and translation on an abstract level, transforming syntactic structures in languages rather than actual meanings.

Often, you will notice that the input alphabet of the automaton or the terminal alphabet of the grammar do not contain actual words themselves, but rather symbols representing word categories and properties. For example, we can use N_{3s} to denote a noun in third person singular. While such representation is sufficient in our examples here, where, for clarity, we usually only focus on some select aspects at a time, in practice we need much more information about each word. In that case, we can, for instance, use structures resembling attribute-value matrices from head-driven phrase structure grammars (see Section 3.1 of the full thesis) as symbols.

4.1 Synchronous Grammars

First, we explore the application perspectives of our newly introduced synchronous grammars, or more precisely, synchronous versions of MATs and SCGs. The original results, observations, and examples presented in this section were published in [21] and [23].

To demonstrate the basic principle, consider a simple Japanese sentence

Takeshi-san wa raishuu Oosaka ni ikimasu.

We will transform this sentence (or, more precisely, the structure of this sentence) into its English counterpart

Takeshi is going to Osaka next week.

In the following examples, words in angled brackets ($\langle\rangle$) are words associated with a terminal or nonterminal symbol in a given sentence or structure. Note that this is included only to make the examples easier to follow and understand, and is not an actual part of the formalism itself.

Example 4.1. Consider a RSCFG $H = (G_I, G_O, \Psi, \varphi_I, \varphi_O)$, where $G_I = (N_I, T_I, P_I, \mathrm{S}_I)$ and $G_O = (N_O, T_O, P_O, \mathrm{S}_O)$ such that

- $N_I = \{\mathrm{S}_I, \text{NP-SBJ}, \text{VP}, \text{PP-TMP}, \text{PP-DIR}\}$,
- $T_I = \{\text{NP}, \text{V}, \text{DET}\}$,
- $N_O = \{\mathrm{S}_O, \text{NP-SBJ}, \text{VP}, \text{PP-TMP}, \text{PP-DIR}\}$,
- $T_O = \{\text{NP}, \text{V}, \text{AUX}, \text{DET}, \text{P}\}$,
- $P_I = \left\{ \begin{array}{rlrl} 1: & \mathrm{S}_I \to \text{NP-SBJ VP}, & 2: & \text{NP-SBJ} \to \text{NP DET}\langle wa \rangle, \\ 3: & \text{VP} \to \text{PP-TMP PP-DIR V}, & 4: & \text{PP-TMP} \to \text{NP}, \\ 4z: & \text{PP-TMP} \to \varepsilon, & 5: & \text{PP-DIR} \to \text{NP DET}\langle ni \rangle, \\ 5z: & \text{PP-DIR} \to \varepsilon & & \end{array} \right\}$,
- $P_O = \left\{ \begin{array}{rlrl} 1: & \mathrm{S}_O \to \text{NP-SBJ VP}, & 2: & \text{NP-SBJ} \to \text{NP}, \\ 3: & \text{VP} \to \text{AUX V PP-DIR PP-TMP}, & 4: & \text{PP-TMP} \to \text{NP}, \\ 4z: & \text{PP-TMP} \to \varepsilon, & 5: & \text{PP-DIR} \to \text{P}\langle to \rangle \text{ NP}, \\ 5z: & \text{PP-DIR} \to \varepsilon & & \end{array} \right\}$.

Strictly according to their definitions, synchronous grammars generate pairs of sentences. However, in practice, we usually have the input sentence in the source language, and we want to translate it into the target language. That is, we want to generate the corresponding output sentence. In that case, the translation can be divided into two steps as follows.

1. First, we parse the input sentence using the input grammar. In G_I, a derivation that generates the example sentence may proceed as follows:

$$\begin{array}{rll} \mathrm{S}_I & \Rightarrow & \text{NP-SBJ VP [1]} \\ & \Rightarrow & \text{NP}\langle \textit{Takeshi-san} \rangle \text{ DET}\langle \textit{wa} \rangle \text{ VP [2]} \\ & \Rightarrow & \text{NP}\langle \textit{Takeshi-san} \rangle \text{ DET}\langle \textit{wa} \rangle \text{ PP-TMP PP-DIR V}\langle \textit{ikimasu} \rangle \text{ [3]} \\ & \Rightarrow & \text{NP}\langle \textit{Takeshi-san} \rangle \text{ DET}\langle \textit{wa} \rangle \text{ NP}\langle \textit{raishuu} \rangle \text{ PP-DIR V}\langle \textit{ikimasu} \rangle \text{ [4]} \\ & \Rightarrow & \text{NP}\langle \textit{Takeshi-san} \rangle \text{ DET}\langle \textit{wa} \rangle \text{ NP}\langle \textit{raishuu} \rangle \text{ NP}\langle \textit{Oosaka} \rangle \text{ DET}\langle \textit{ni} \rangle \\ & & \text{V}\langle \textit{ikimasu} \rangle \text{ [5]} \end{array}$$

We have applied rules denoted by labels 1 2 3 4 5, in that order.

2. Next, we use the sequence obtained in the first step (1 2 3 4 5), and apply the corresponding rules in the output grammar. Then, the derivation in G_O proceeds as follows:

$$\begin{aligned}
\mathrm{S}_O &\Rightarrow \text{NP-SBJ VP [1]} \\
&\Rightarrow \text{NP}\langle \textit{Takeshi} \rangle \text{ VP [2]} \\
&\Rightarrow \text{NP}\langle \textit{Takeshi} \rangle \text{ AUX}\langle \textit{is} \rangle \text{ V}\langle \textit{going} \rangle \text{ PP-TMP PP-DIR [3]} \\
&\Rightarrow \text{NP}\langle \textit{Takeshi} \rangle \text{ AUX}\langle \textit{is} \rangle \text{ V}\langle \textit{going} \rangle \text{ PP-DIR NP}\langle \textit{next week} \rangle \text{ [4]} \\
&\Rightarrow \text{NP}\langle \textit{Takeshi} \rangle \text{ AUX}\langle \textit{is} \rangle \text{ V}\langle \textit{going} \rangle \text{ P}\langle \textit{to} \rangle \text{ NP}\langle \textit{Osaka} \rangle \text{ NP}\langle \textit{next week} \rangle \text{ [5]}
\end{aligned}$$

Also note the rules $4z$ and $5z$ (in both input and outpur grammar), which can be used to erase PP-TMP and PP-DIR. This represents the fact that these constituents may be omitted.

The full thesis further elaborates upon this example, demonstrating different grammatical categories and syntactic structures.

Let us now consider translation between Czech and English. Czech is a relatively challenging language in terms of natural language processing. It is a free-word-order language with rich inflection (see [16]).

For example, consider the Czech sentence

Dva růžoví sloni přišli na přednášku.
(Two pink elephants came to the lecture.)

All of the following permutations of words also make for a valid sentence:

dva růžoví sloni přišli na přednášku	*dva růžoví sloni na přednášku přišli*
růžoví sloni přišli na přednášku dva	*růžoví sloni na přednášku přišli dva*
dva sloni přišli na přednášku růžoví	*dva sloni na přednášku přišli růžoví*
sloni přišli na přednášku dva růžoví	*sloni na přednášku přišli dva růžoví*

There may be differences in meaning or emphasis, but the syntactic structure remains the same. Why is this problematic? Compare the syntax trees in Figure 4.1. Because of the crossing branches (non-projectivity), the second tree cannot be produced by any CFG. Of course, it is still possible to construct a CFG that generates the sentence *růžoví sloni přišli na přednášku dva* if we consider a different syntax tree, for example such as in Figure 4.2. However, this tree no longer captures the relation between the noun *sloni* and its modifying numeral *dva* (represented by the dotted line). We need to know this relation for instance to ensure agreement between the words (person, number, gender...), so that we can choose their appropriate forms.

In a purely context-free framework, this can be complicated. The necessary information has to be propagated through the derivation tree, even if the structure is not actually affected, and this can result in a high number of rules. Recall that in generalized phrase structure grammars, for instance, this is countered by the introduction of metarules and features (see Section 3.1 of the full thesis). With MATs, we can instead represent the relations using matrices.

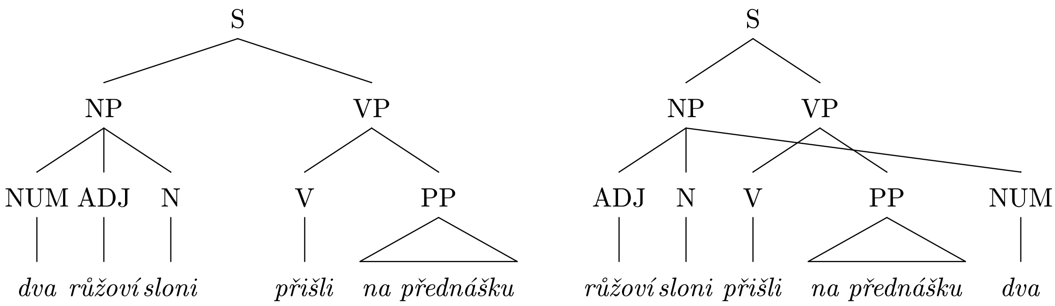

Figure 4.1: Syntax trees for example sentences in Czech

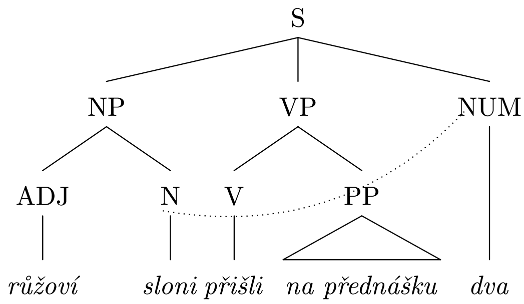

Figure 4.2: Modified syntax tree

Example 4.2. Here, we present an example of SMAT $H = (G_{cz}, M_{cz}, G_{en}, M_{en}, \Psi, \varphi_{cz}, \varphi_{en})$ that describes the translations between the English sentence *two pink elephants came to the lecture* and any of the above Czech sentences, correctly distinguishing between male and female gender in Czech (to demonstrate female gender, we also include *opice* in Czech, *monkeys* in English). Note that H is actually more general (for example allowing multiple adjectives). It is designed for easy extension to include other grammatical categories (person…) as well as different syntactic structures.

For Czech, let G_{cz} contain the following context-free rules (nonterminals are in capitals, S_{cz} is the start symbol):

$$\begin{array}{rl@{\qquad}rl}
\mathrm{s}: & \mathrm{S}_{cz} \to \mathrm{NP\ VP\ NUM\ ADJS}, & \mathrm{np}: & \mathrm{NP} \to \mathrm{NUM\ ADJS\ N}, \\
\mathrm{vp}: & \mathrm{VP} \to \mathrm{ADVS\ V\ ADVS}, & \mathrm{num}_\varepsilon: & \mathrm{NUM} \to \varepsilon, \\
\mathrm{adjs}: & \mathrm{ADJS} \to \mathrm{ADJ\ ADJS}, & \mathrm{adjs}_\varepsilon: & \mathrm{ADJS} \to \varepsilon, \\
\mathrm{advs}: & \mathrm{ADVS} \to \mathrm{ADV\ ADVS}, & \mathrm{advs}_\varepsilon: & \mathrm{ADVS} \to \varepsilon, \\
\mathrm{n}_m: & \mathrm{N} \to \mathrm{N}_m, & \mathrm{n}_f: & \mathrm{N} \to \mathrm{N}_f, \\
\mathrm{n}_{mm}: & \mathrm{N}_m \to \mathrm{N}_m, & \mathrm{n}_{ff}: & \mathrm{N}_f \to \mathrm{N}_f, \\
\mathrm{v}_m: & \mathrm{V} \to \mathrm{V}_m, & \mathrm{v}_f: & \mathrm{V} \to \mathrm{V}_f, \\
\mathrm{adj}_m: & \mathrm{ADJ} \to \mathrm{ADJ}_m, & \mathrm{adj}_f: & \mathrm{ADJ} \to \mathrm{ADJ}_f, \\
\mathrm{adv}: & \mathrm{ADV} \to \mathrm{PP}, & \mathrm{num}_m: & \mathrm{NUM} \to \mathrm{NUM}_m, \\
\mathrm{num}_f: & \mathrm{NUM} \to \mathrm{NUM}_f, & \mathrm{dict}_1: & \mathrm{N}_m \to \textit{sloni}, \\
\mathrm{dict}_2: & \mathrm{N}_f \to \textit{opice}, & \mathrm{dict}_{3m}: & \mathrm{V}_m \to \textit{přišli}, \\
\mathrm{dict}_{3f}: & \mathrm{V}_f \to \textit{přišly}, & \mathrm{dict}_{4m}: & \mathrm{ADJ}_m \to \textit{růžoví}, \\
\mathrm{dict}_{4f}: & \mathrm{ADJ}_f \to \textit{růžové}, & \mathrm{dict}_{5m}: & \mathrm{NUM}_m \to \textit{dva}, \\
\mathrm{dict}_{5f}: & \mathrm{NUM}_f \to \textit{dvě}, & \mathrm{dict}_6: & \mathrm{PP} \to \textit{na přednášku}
\end{array}$$

Similarly, for English, let G_{en} contain the following rules (again, nonterminals are in capitals, and S_{en} is the start symbol):

s :	$S_{en} \to$ NP VP,	np :	NP $\to$ NUM ADJS N,
vp :	VP $\to$ V ADVS,	num_ε :	NUM $\to \varepsilon$,
adjs :	ADJS $\to$ ADJ ADJS,	adjs_ε :	ADJS $\to \varepsilon$,
advs :	ADVS $\to$ ADV ADVS,	advs_ε :	ADVS $\to \varepsilon$,
adv :	ADV $\to$ PP,	dict_1 :	N $\to$ *elephants*,
dict_2 :	N $\to$ *monkeys*,	dict_3 :	V $\to$ *came*,
dict_4 :	ADJ $\to$ *pink*,	dict_5 :	NUM $\to$ *two*,
dict_6 :	PP $\to$ *to the lecture*		

Finally, let M_{cz} and M_{en} contain the following matrices:

	M_{cz}	M_{en}		M_{cz}	M_{en}
s :	s	s	np :	np	np
vp :	vp	vp	num :	num_ε	ε
num_ε :	num_ε num_ε	num_ε	$adjs$:	adjs	adjs
$adjs_\varepsilon$:	adjs_ε adjs_ε	adjs_ε	$advs$:	advs	advs
$advs_\varepsilon$:	advs_ε advs_ε	advs_ε	n_m :	n_m	ε
n_f :	n_f	ε	v_m :	v_m n_{mm}	ε
v_f :	v_f n_{ff}	ε	adj_m :	adj_m n_{mm}	ε
adj_f :	adj_f n_{ff}	ε	adv :	adv	adv
num_m :	num_m n_{mm}	ε	num_f :	num_f n_{ff}	ε
$dict_1$:	dict_1	dict_1	$dict_2$:	dict_2	dict_2
$dict_{3m}$:	dict_{3m}	dict_3	$dict_{3f}$:	dict_{3f}	dict_3
$dict_{4m}$:	dict_{4m}	dict_4	$dict_{4f}$:	dict_{4f}	dict_4
$dict_{5m}$:	dict_{5m}	dict_5	$dict_{5f}$:	dict_{5f}	dict_5
$dict_6$:	dict_6	dict_6			

In this example, we have chosen to include the words themselves directly in the grammar rules (rather than assuming a separate dictionary) to illustrate this approach as well. For instance, consider the rule dict_{5m} in G_{cz}. This rule encodes the fact that the word *dva* (in Czech) is a numeral, of male gender (in practice, there can be much more information). We call this kind of rules dictionary rules.

Further, note for example the matrix adj_f in M_{cz}, which ensures agreement between noun and adjective (both must be in female gender). Another interesting matrix is $adjs_\varepsilon$, which terminates generation of adjectives. In the Czech sentence in this example, we have two positions where adjectives can be placed (directly within the noun phrase or at the end of the sentence). In English, there is only one possible position (within the noun phrase). This is why the rule ADJS $\to \varepsilon$ is used twice in Czech, but only once in English.

Also observe that the linked matrices (sharing the same label) in M_{cz} and M_{en} may contain completely different rules and they can even be empty (ε), in which case the corresponding grammar does not change its sentential form in that step. The definitions of MAT and SMAT allow for this kind of flexibility when describing both individual languages and their translations.

Example of a derivation in Czech follows.

$$
\begin{aligned}
\mathrm{S}_{cz} &\Rightarrow \text{NP VP NUM ADJS } [s] \\
&\Rightarrow \text{NUM ADJS N VP NUM ADJS } [np] \\
&\Rightarrow \text{NUM ADJS N ADVS V ADVS NUM ADJS } [vp] \\
&\Rightarrow \text{ADJS N ADVS V ADVS NUM ADJS } [num] \\
&\Rightarrow \text{ADJ ADJS N ADVS V ADVS NUM ADJS } [adjs] \\
&\Rightarrow \text{ADJ N ADVS V ADVS NUM } [adjs_\varepsilon] \\
&\Rightarrow \text{ADJ N ADVS V ADV ADVS NUM } [advs] \\
&\Rightarrow \text{ADJ N V ADV NUM } [advs_\varepsilon] \\
&\Rightarrow \text{ADJ } \mathrm{N}_m \text{ V ADV NUM } [n_m] \\
&\Rightarrow \text{ADJ } \mathrm{N}_m\ \mathrm{V}_m \text{ ADV NUM } [v_m] \\
&\Rightarrow \mathrm{ADJ}_m\ \mathrm{N}_m\ \mathrm{V}_m \text{ ADV NUM } [adj_m] \\
&\Rightarrow \mathrm{ADJ}_m\ \mathrm{N}_m\ \mathrm{V}_m \text{ PP NUM } [adv] \\
&\Rightarrow \mathrm{ADJ}_m\ \mathrm{N}_m\ \mathrm{V}_m \text{ PP } \mathrm{NUM}_m\ [num_m] \\
&\Rightarrow \mathrm{ADJ}_m\ \textit{sloni}\ \mathrm{V}_m \text{ PP } \mathrm{NUM}_m\ [dict_1] \\
&\Rightarrow \mathrm{ADJ}_m\ \textit{sloni přišli} \text{ PP } \mathrm{NUM}_m\ [dict_{3m}] \\
&\Rightarrow \textit{růžoví sloni přišli} \text{ PP } \mathrm{NUM}_m\ [dict_{4m}] \\
&\Rightarrow \textit{růžoví sloni přišli na přednášku}\ \mathrm{NUM}_m\ [dict_5] \\
&\Rightarrow \textit{růžoví sloni přišli na přednášku dva}\ [dict_{6m}]
\end{aligned}
$$

The corresponding derivation in English may look like this:

$$
\begin{aligned}
\mathrm{S}_{en} &\Rightarrow \text{NP VP } [s] \\
&\Rightarrow \text{NUM ADJS N VP } [np] \\
&\Rightarrow \text{NUM ADJS N V ADVS } [vp] \\
&\Rightarrow \text{NUM ADJS N V ADVS } [num] \\
&\Rightarrow \text{NUM ADJ ADJS N V ADVS } [adjs] \\
&\Rightarrow \text{NUM ADJ N V ADVS } [adjs_\varepsilon] \\
&\Rightarrow \text{NUM ADJ N V ADV ADVS } [advs] \\
&\Rightarrow \text{NUM ADJ N V ADV } [advs_\varepsilon] \\
&\Rightarrow \text{NUM ADJ N V ADV } [n_m] \\
&\Rightarrow \text{NUM ADJ N V ADV } [v_m] \\
&\Rightarrow \text{NUM ADJ N V ADV } [adj_m] \\
&\Rightarrow \text{NUM ADJ N V PP } [adv] \\
&\Rightarrow \text{NUM ADJ N V PP } [num_m] \\
&\Rightarrow \text{NUM ADJ } \textit{elephants}\ \mathrm{V}_m \text{ PP } [dict_1] \\
&\Rightarrow \text{NUM ADJ } \textit{elephants came} \text{ PP } [dict_{3m}] \\
&\Rightarrow \text{NUM } \textit{pink elephants came} \text{ PP } [dict_{4m}] \\
&\Rightarrow \text{NUM } \textit{pink elephants came to the lecture}\ [dict_5] \\
&\Rightarrow \textit{two pink elephants came to the lecture}\ [dict_{6m}]
\end{aligned}
$$

The entire derivation tree for the Czech sentence is shown in Figure 4.3. The dotted lines represent relations described by matrices. The triangle from N_m to N_m is an abstraction which in this particular case essentially means that this step is repeated until all agreement issues are resolved.

We can achieve similar results using SSCGs. For example the matrix adj_f in M_{cz} can be represented by two scattered-context rules $(\mathrm{ADJ}, \mathrm{N}_f) \rightarrow (\mathrm{ADJ}_f, \mathrm{N}_f)$ and $(\mathrm{N}_f, \mathrm{ADJ}) \rightarrow (\mathrm{N}_f, \mathrm{ADJ}_f)$. Note that we need two rules, because the nonterminal order is important in SCG (this is one of the key differences between SMAT and SSCG). In this case, we need an additional rule in SSCG. However, this can also be an advantage, because it allows us to easily distinguish between left and right modifiers. For example, if we only have the first

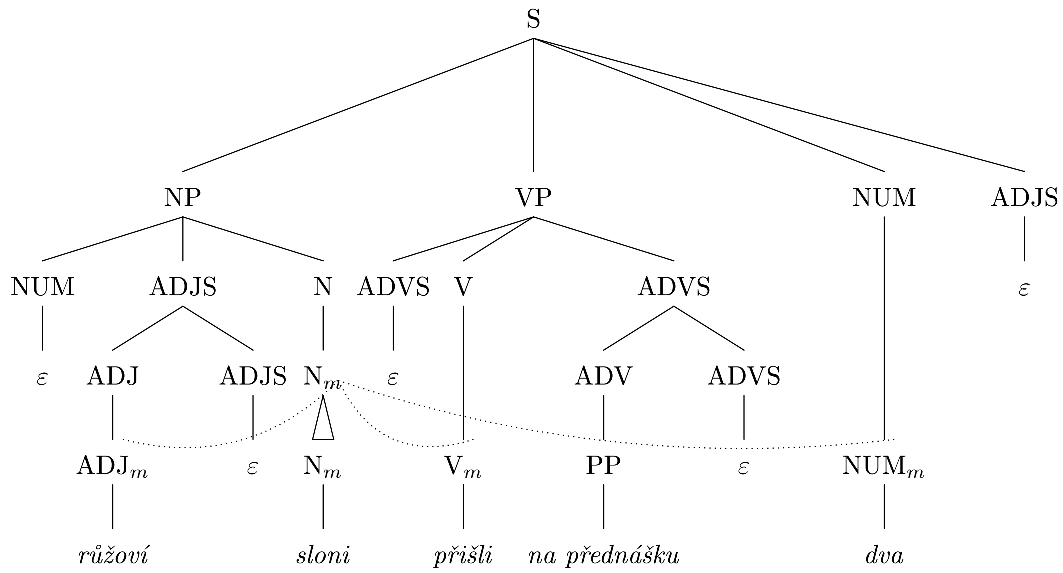

Figure 4.3: Derivation tree of G_{cz}

rule (ADJ, N_f) $\rightarrow$ (ADJ_f, N_f), it means that the adjective always has to occur on the left of the noun.

4.2 Rule-Restricted Transducers

In this section, we discuss the applications perspectives of RTs. The original results, observations, and examples presented here were first published in [6].

The thesis includes several case studies, the first of which demonstrates the basic principles of RTs using a simple example of passive transformation in English. Subsequent examples consider some of the more complex features and structures of natural languages.

Here, we present one of the examples dealing with translation between different languages. We focus on Japanese, Czech, and English. One problem when translating into Czech is that there is very rich inflection and the form of the words reflects many grammatical categories, such as case, gender, or number (see [16], where the author discusses this issue with regard to computational linguistics). To illustrate, compare the following sentences in Japanese, English, and Czech.

Zasshi o yondeitta onna no hito wa watashi no shiriai deshita.
Zasshi o yondeitta otoko no hito wa watashi no shiriai deshita.

The woman who was reading a magazine was an acquitance of mine.
The man who was reading a magazine was an acquitance of mine.

Žena, která četla časopis, byla moje známá.
Muž, který četl časopis, byl můj známý.

As we can see, in Czech, nearly every word is different, depending on the gender of the subject. In contrast, in both Japanese and English, the two sentences only differ in one word, namely *onna no hito* (*woman*) and *otoko no hito* (*man*).[1]

The above sentences also give us an example of some structural differences between Japanese and Czech. In Czech and English, the structure of the sentence is very similar, but in Japanese, there is no word that correspond directly to *který* (*which*, *who*, ...). Instead, this relation is represented by the form of the verb *yondeitta* (the dictionary form is *yomu*, meaning *to read*). Compare the syntax trees in Figure 4.4.

Example 4.3. Consider an RT $\Gamma = (M, G, \Psi)$, where $M = (Q, \Sigma, \delta, 0, F)$, $G = (N, T, P, \mathrm{S})$ such that

- $Q = \{0, m, m1, m2, f, f1, f2, n, n1, n2, 1m, 1f, 1n\}$,
- $\Sigma = \{\mathrm{NP}_m,\ \mathrm{NP}_f,\ \mathrm{NP}_n,\ \mathrm{NP}_?,\ \mathrm{V},\ \mathrm{DET},\ \#\}$,
- $F = \{1m, 1f, 1n\}$,
- $N = \{\mathrm{S},\ \mathrm{S}',\ \text{NP-SBJ},\ \mathrm{NP}_?,\ \mathrm{VP},\ \mathrm{PN}_?,\ \mathrm{V}_?,\ \mathrm{X}\}$,
- $T = \{\mathrm{NP}_m,\ \mathrm{NP}_f,\ \mathrm{NP}_n,\ \mathrm{V}_m,\ \mathrm{V}_f,\ \mathrm{V}_n,\ \mathrm{PN}_m,\ \mathrm{PN}_f,\ \mathrm{PN}_n\}$,
- $$\delta = \left\{ \begin{array}{rlrlrl} r_1: & 0\mathrm{V} \to 1, & r_2: & 1 \to 0, & r_3: & 0\mathrm{NP}_? \to 0, \\ r_4: & 0\mathrm{DET} \to 0, & r_{5m}: & 0\mathrm{NP}_m \to m, & r_{5f}: & 0\mathrm{NP}_f \to f, \\ r_{5n}: & 0\mathrm{NP}_n \to n, & r_{m1}: & m\mathrm{V} \to m1, & r_{m2}: & m1 \to m2, \\ r_{m3}: & m2 \to m, & r_{m4}: & m\mathrm{DET} \to m, & r_{m5}: & m\mathrm{NP}_? \to m, \\ r_{m5m}: & m\mathrm{NP}_m \to m, & r_{m5f}: & m\mathrm{NP}_f \to m, & r_{m5n}: & m\mathrm{NP}_n \to m, \\ r_{m6}: & m\# \to 1m, & r_{m7}: & 1m \to 1m, & r_{f1}: & f\mathrm{V} \to f1, \\ r_{f2}: & f1 \to f2, & \ldots & & r_{f7}: & 1f \to 1f, \\ r_{n1}: & n\mathrm{V} \to n1, & r_{n2}: & n1 \to n2, & \ldots & \\ r_{n7}: & 1n \to 1n & & & & \end{array} \right\},$$
- $$P = \left\{ \begin{array}{rlrl} p_1: & \mathrm{S} \to \text{NP-SBJ VP X}, & p_2: & \text{NP-SBJ} \to \mathrm{NP}_?, \\ p_{3m}: & \mathrm{NP}_? \to \mathrm{NP}_m, & p_{3f}: & \mathrm{NP}_? \to \mathrm{NP}_f, \\ p_{3n}: & \mathrm{NP}_? \to \mathrm{NP}_n, & p_4: & \mathrm{NP}_? \to \mathrm{NP}_?, \\ p_5: & \mathrm{NP}_? \to \mathrm{NP}_?\ \mathrm{S}', & p_6: & \mathrm{VP} \to \mathrm{V}_?\ \mathrm{NP}_?, \\ p_{7m}: & \mathrm{V}_? \to \mathrm{V}_m, & p_{7f}: & \mathrm{V}_? \to \mathrm{V}_f, \\ p_{7n}: & \mathrm{V}_? \to \mathrm{V}_n, & p_8: & \mathrm{S}' \to \mathrm{PN}_?\ \mathrm{VP}, \\ p_{9m}: & \mathrm{PN}_? \to \mathrm{PN}_m, & p_{9f}: & \mathrm{PN}_? \to \mathrm{PN}_f, \\ p_{9n}: & \mathrm{PN}_? \to \mathrm{PN}_n, & p_{10}: & \mathrm{X} \to \varepsilon \end{array} \right\},$$
- $\Psi = \{(r_1, p_1), (r_2, p_6), (r_3, p_4), (r_4, p_2), (r_{5m}, p_4), (r_{5f}, p_4), (r_{5n}, p_4), (r_{m1}, p_5), (r_{m2}, p_8), (r_{m3}, p_6), (r_{m4}, p_4), (r_{m5}, p_4), (r_{m5m}, p_{3m}), (r_{m5f}, p_{3f}), (r_{m5n}, p_{3n}), (r_{m6}, p_{10}), (r_{m7}, p_{3m}), (r_{m7}, p_{7m}), (r_{m7}, p_{9m}), (r_{f1}, p_5), (r_{f2}, p_8), (r_{f3}, p_6), (r_{f4}, p_4), (r_{f5}, p_4), (r_{f5m}, p_{3m}), (r_{f5f}, p_{3f}), (r_{f5n}, p_{3n}), (r_{f6}, p_{10}), (r_{f7}, p_{3f}), (r_{f7}, p_{7f}), (r_{f7}, p_{9f}), (r_{n1}, p_5), (r_{n2}, p_8), (r_{n3}, p_6), (r_{n4}, p_4), (r_{n5}, p_4), (r_{n5m}, p_{3m}), (r_{n5f}, p_{3f}), (r_{n5n}, p_{3n}), (r_{n6}, p_{10}), (r_{n7}, p_{3n}), (r_{n7}, p_{7n}), (r_{n7}, p_{9n})\}$.

We have added two dummy symbols: the input symbol #, which acts as the endmarker, and the nonterminal X, which we generate at the beginning of the computation and then erase when all the input has been read (including #).

[1] Technically, *onna no hito* literally translates to *woman's person* or *female person*, with *onna* itself meaning *woman*, *female*. However, referring to a person only by *onna* may have negative connotations in Japanese. This is analogous for *otoko no hito*.

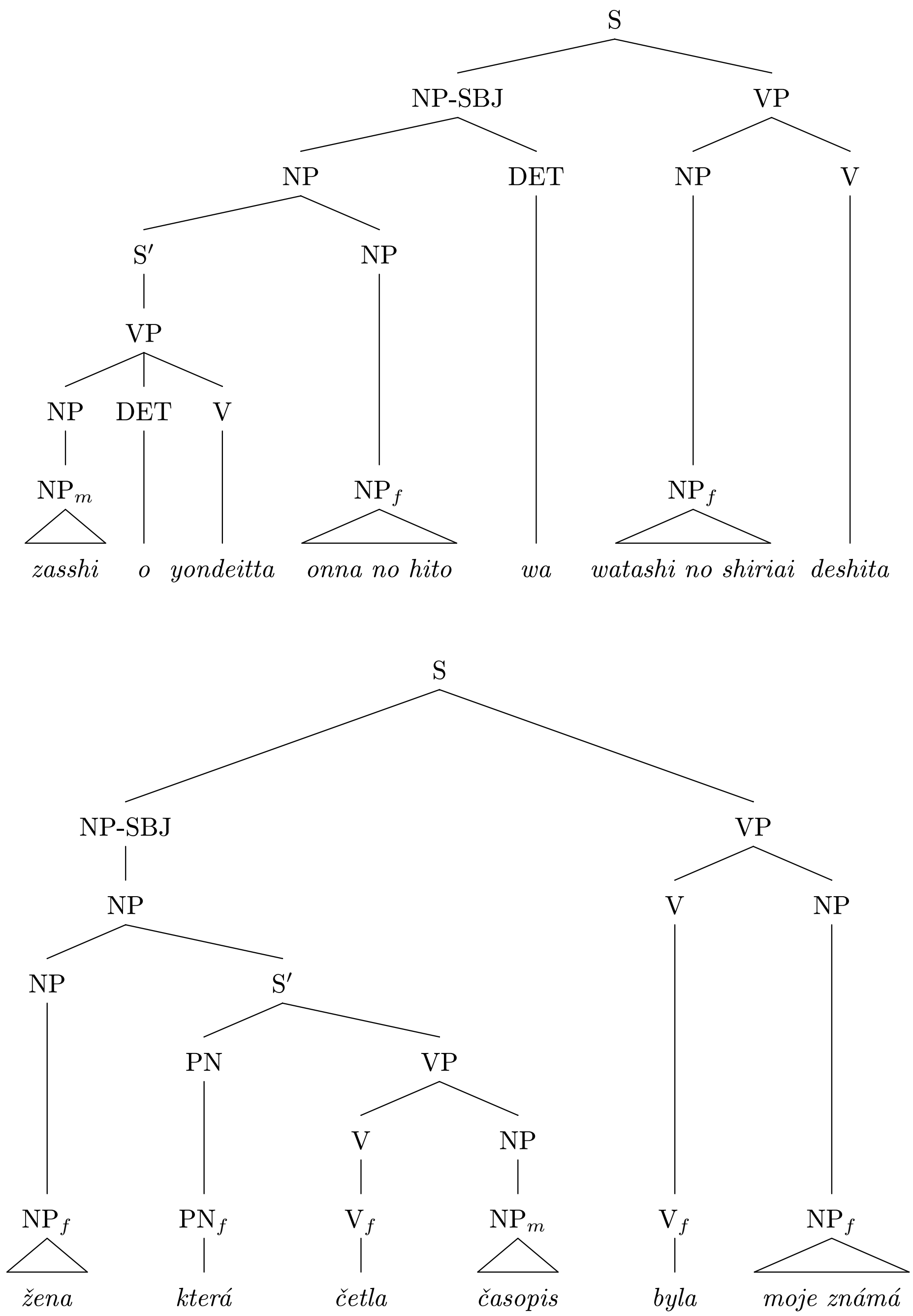

Figure 4.4: Syntax trees for Japanese (top) and Czech sentence

In this example, we read the input sentence in reverse order (right to left). Clearly, this makes no difference from a purely theoretical point of view, but it can be more suitable in practice due to the way how Japanese sentences are organized.

The computation transforming the sentence *zasshi o yondeitta onna no hito wa watashi no shiriai deshita* into *žena, která četla časopis, byla moje známá* can proceed as follows:

$$
\begin{array}{ll}
 & (0\ \underline{\mathrm{V}\langle \textit{deshita}\rangle}\ \mathrm{NP}_{?}\langle \textit{watashi no shiriai}\rangle\ \mathrm{DET}\langle \textit{wa}\rangle\ \mathrm{NP}_{f}\langle \textit{onna no hito}\rangle \\
 & \mathrm{V}\langle \textit{yondeitta}\rangle\ \mathrm{DET}\langle \textit{o}\rangle\ \mathrm{NP}_{m}\langle \textit{zasshi}\rangle\ \#,\ \underline{\mathrm{S}}) \\
\Rightarrow & (1\ \mathrm{NP}_{?}\langle \textit{watashi no shiriai}\rangle\ \mathrm{DET}\langle \textit{wa}\rangle\ \mathrm{NP}_{f}\langle \textit{onna no hito}\rangle\ \mathrm{V}\langle \textit{yondeitta}\rangle \\
 & \mathrm{DET}\langle \textit{o}\rangle\ \mathrm{NP}_{m}\langle \textit{zasshi}\rangle\ \#,\ \text{NP-SBJ}\ \underline{\mathrm{VP}}\ \mathrm{X})\ [(r_1, p_1)] \\
\Rightarrow & (0\ \underline{\mathrm{NP}_{?}\langle \textit{watashi no shiriai}\rangle}\ \mathrm{DET}\langle \textit{wa}\rangle\ \mathrm{NP}_{f}\langle \textit{onna no hito}\rangle\ \mathrm{V}\langle \textit{yondeitta}\rangle \\
 & \mathrm{DET}\langle \textit{o}\rangle\ \mathrm{NP}_{m}\langle \textit{zasshi}\rangle\ \#,\ \text{NP-SBJ}\ \mathrm{V}_{?}\langle \textit{byl}\rangle\ \underline{\mathrm{NP}_{?}}\ \mathrm{X})\ [(r_2, p_6)] \\
\Rightarrow & (0\ \underline{\mathrm{DET}\langle \textit{wa}\rangle}\ \mathrm{NP}_{f}\langle \textit{onna no hito}\rangle\ \mathrm{V}\langle \textit{yondeitta}\rangle\ \mathrm{DET}\langle \textit{o}\rangle\ \mathrm{NP}_{m}\langle \textit{zasshi}\rangle\ \#, \\
 & \underline{\text{NP-SBJ}}\ \mathrm{V}_{?}\langle \textit{byl}\rangle\ \mathrm{NP}_{?}\langle \textit{můj známý}\rangle\ \mathrm{X})\ [(r_3, p_4)] \\
\Rightarrow & (0\ \underline{\mathrm{NP}_{f}\langle \textit{onna no hito}\rangle}\ \mathrm{V}\langle \textit{yondeitta}\rangle\ \mathrm{DET}\langle \textit{o}\rangle\ \mathrm{NP}_{m}\langle \textit{zasshi}\rangle\ \#,\ \underline{\mathrm{NP}_{?}} \\
 & \mathrm{V}_{?}\langle \textit{byl}\rangle\ \mathrm{NP}_{?}\langle \textit{můj známý}\rangle\ \mathrm{X})\ [(r_4, p_2)] \\
\Rightarrow & (f\ \underline{\mathrm{V}\langle \textit{yondeitta}\rangle}\ \mathrm{DET}\langle \textit{o}\rangle\ \mathrm{NP}_{m}\langle \textit{zasshi}\rangle\ \#,\ \underline{\mathrm{NP}_{?}\langle \textit{žena}\rangle}\ \mathrm{V}_{?}\langle \textit{byl}\rangle \\
 & \mathrm{NP}_{?}\langle \textit{můj známý}\rangle\ \mathrm{X})\ [(r_{5f}, p_4)] \\
\Rightarrow & (f1\ \mathrm{DET}\langle \textit{o}\rangle\ \mathrm{NP}_{m}\langle \textit{zasshi}\rangle\ \#,\ \mathrm{NP}_{?}\langle \textit{žena}\rangle\ \underline{\mathrm{S}'}\ \mathrm{V}_{?}\langle \textit{byl}\rangle\ \mathrm{NP}_{?}\langle \textit{můj známý}\rangle\ \mathrm{X}) \\
 & [(r_{f1}, p_5)] \\
\Rightarrow & (f2\ \mathrm{DET}\langle \textit{o}\rangle\ \mathrm{NP}_{m}\langle \textit{zasshi}\rangle\ \#,\ \mathrm{NP}_{?}\langle \textit{žena}\rangle\ \mathrm{PN}_{?}\langle \textit{který}\rangle\ \underline{\mathrm{VP}}\ \mathrm{V}_{?}\langle \textit{byl}\rangle \\
 & \mathrm{NP}_{?}\langle \textit{můj známý}\rangle\ \mathrm{X})\ [(r_{f2}, p_8)] \\
\Rightarrow & (f\ \underline{\mathrm{DET}\langle \textit{o}\rangle}\ \mathrm{NP}_{m}\langle \textit{zasshi}\rangle\ \#,\ \mathrm{NP}_{?}\langle \textit{žena}\rangle\ \mathrm{PN}_{?}\langle \textit{který}\rangle\ \mathrm{V}_{?}\langle \textit{četl}\rangle\ \underline{\mathrm{NP}_{?}}\ \mathrm{V}_{?}\langle \textit{byl}\rangle \\
 & \mathrm{NP}_{?}\langle \textit{můj známý}\rangle\ \mathrm{X})\ [(r_{f3}, p_6)] \\
\Rightarrow & (f\ \underline{\mathrm{NP}_{m}\langle \textit{zasshi}\rangle}\ \#,\ \mathrm{NP}_{?}\langle \textit{žena}\rangle\ \mathrm{PN}_{?}\langle \textit{který}\rangle\ \mathrm{V}_{?}\langle \textit{četl}\rangle\ \underline{\mathrm{NP}_{?}}\ \mathrm{V}_{?}\langle \textit{byl}\rangle \\
 & \mathrm{NP}_{?}\langle \textit{můj známý}\rangle\ \mathrm{X})\ [(r_{f4}, p_4)] \\
\Rightarrow & (f\ \underline{\#},\ \mathrm{NP}_{?}\langle \textit{žena}\rangle\ \mathrm{PN}_{?}\langle \textit{který}\rangle\ \mathrm{V}_{?}\langle \textit{četl}\rangle\ \mathrm{NP}_{m}\langle \textit{časopis}\rangle\ \mathrm{V}_{?}\langle \textit{byl}\rangle \\
 & \mathrm{NP}_{?}\langle \textit{můj známý}\rangle\ \underline{\mathrm{X}})\ [(r_{f5m}, p_{3m})] \\
\Rightarrow & (1f,\ \underline{\mathrm{NP}_{?}\langle \textit{žena}\rangle}\ \mathrm{PN}_{?}\langle \textit{který}\rangle\ \mathrm{V}_{?}\langle \textit{četl}\rangle\ \mathrm{NP}_{m}\langle \textit{časopis}\rangle\ \mathrm{V}_{?}\langle \textit{byl}\rangle \\
 & \mathrm{NP}_{?}\langle \textit{můj známý}\rangle)\ [(r_{f6}, p_{10})] \\
\Rightarrow & (1f,\ \mathrm{NP}_{f}\langle \textit{žena}\rangle\ \underline{\mathrm{PN}_{?}\langle \textit{který}\rangle}\ \mathrm{V}_{?}\langle \textit{četl}\rangle\ \mathrm{NP}_{m}\langle \textit{časopis}\rangle\ \mathrm{V}_{?}\langle \textit{byl}\rangle \\
 & \mathrm{NP}_{?}\langle \textit{můj známý}\rangle)\ [(r_{f7}, p_{3f})] \\
\Rightarrow & (1f,\ \mathrm{NP}_{f}\langle \textit{žena}\rangle\ \mathrm{PN}_{f}\langle \textit{která}\rangle\ \underline{\mathrm{V}_{?}\langle \textit{četl}\rangle}\ \mathrm{NP}_{m}\langle \textit{časopis}\rangle\ \mathrm{V}_{?}\langle \textit{byl}\rangle \\
 & \mathrm{NP}_{?}\langle \textit{můj známý}\rangle)\ [(r_{f7}, p_{9f})] \\
\Rightarrow & (1f,\ \mathrm{NP}_{f}\langle \textit{žena}\rangle\ \mathrm{PN}_{f}\langle \textit{která}\rangle\ \mathrm{V}_{f}\langle \textit{četla}\rangle\ \mathrm{NP}_{m}\langle \textit{časopis}\rangle\ \underline{\mathrm{V}_{?}\langle \textit{byl}\rangle} \\
 & \mathrm{NP}_{?}\langle \textit{můj známý}\rangle)\ [(r_{f7}, p_{7f})] \\
\Rightarrow & (1f,\ \mathrm{NP}_{f}\langle \textit{žena}\rangle\ \mathrm{PN}_{f}\langle \textit{která}\rangle\ \mathrm{V}_{f}\langle \textit{četla}\rangle\ \mathrm{NP}_{m}\langle \textit{časopis}\rangle\ \mathrm{V}_{f}\langle \textit{byla}\rangle \\
 & \underline{\mathrm{NP}_{?}\langle \textit{můj známý}\rangle})\ [(r_{f7}, p_{7f})] \\
\Rightarrow & (1f,\ \mathrm{NP}_{f}\langle \textit{žena}\rangle\ \mathrm{PN}_{f}\langle \textit{která}\rangle\ \mathrm{V}_{f}\langle \textit{četla}\rangle\ \mathrm{NP}_{m}\langle \textit{časopis}\rangle\ \mathrm{V}_{f}\langle \textit{byla}\rangle \\
 & \mathrm{NP}_{f}\langle \textit{moje známá}\rangle)\ [(r_{f7}, p_{3f})]
\end{array}
$$

When we first read the word that determines the gender, we move to the state that represents this gender (state m, n, or f). Note that these states are functionally identical in the sense that we can read the same input symbols, while performing the same computation steps in the grammar generating the output. After we have reached the end of input, we rewrite the nonterminal symbols representing words with as of yet unknown gender to the corresponding terminal symbols, depending on the state.

4.3 Summary

In the first two sections of this chapter, we have tried to point out and illustrate the key advantages of the proposed formal models using select case studies from the Czech, English, and Japanese language. Here, we summarize them, and compare the respective strengths and weaknesses of the new synchronous grammars and RTs. The observations presented are based on our previously published papers [6], [21], and [23].

One of the main advantages of both types of models is their power. As shown in Chapters 2 and 3, both synchronous grammars (with linked rules) and RTs (without leftmost restriction) are able to describe even some non-context-free languages. Although arguably relatively rare in practice, there are some features of natural languages that are difficult or impossible to properly capture with CFGs only (such as cross-dependencies). Furthermore, even in cases when a purely context-free description is possible, it may require a high number of rules. Our new models can provide a more economical description thanks to their increased generative power and, in case of RTs, also accepting power.

Another advantage of our new synchronous grammars is their high flexibility, especially if we synchronize models that have higher generative power themselves, such as regulated grammars. In particular, let us consider the case of SMAT. As shown above (Theorem 2.5), if we synchronize MATs in the proposed fashion, we do not obtain any further increase in power of the whole system compared to RSCFG or MAT. However, more powerful individual components allow for easier—and again, more economical—description of each individual language.

Unlike synchronous grammars, which are symmetric and therefore can be used for bidirectional translation, RTs can only describe translation in one direction. Furthermore, because their components are relatively simple (an FA and a CFG), RTs are also less flexible than, for example, SMATs and SSCGs. Consequently, the description of linguistic structures and features can be more complex (essentially, requiring more rules).

On the other hand, the simplicity of components can also be seen as an important advantage of RT, especially from a practical viewpoint. Both FAs and CFGs are well-known and well-studied not only from a theoretical point of view, but also with regards to practical implementations. For example, there are well-known methods of efficient parsing for CFGs.

Another advantage of RT lies in its the straightforward and intuitive basic principle (read input with an FA, generate output with a CFG), which directly corresponds to the translation task in practice. In contrast, in synchronous grammars, both components generate sentences.

Finally, note that both types of introduced formal models can be extended for use in statistical natural processing as well. We can, for example, assign weights (or probabilities) to rules similarly to probabilistic CFGs or weighted synchronous grammars.

Chapter 5

Conclusion

In this doctoral thesis, we have presented new grammar systems that can formally describe translations (or, more specifically, transformations of syntactic structures). We have discussed some of the theoretical properties of the new models, in particular their generative and accepting power.

More specifically, we have introduced the idea of synchronization based on linked rules as a modification of the well-known synchronous grammars. We have extended this principle beyond CFGs, to models with regulated rewriting, defining sychronous MATs and synchronous SCGs.

Further, we have introduced the rule-restricted automaton-grammar transducer, based on the natural idea of reading some input with an FA and producing an appropriate output with a CFG, and provided precise formal definitons. We have also considered two of its variants, namely leftmost restricted RTs and RTs with appearance checking.

We have established the following main results:

1. Rule-synchronized CFGs are more powerful than CFGs, as they characterize the same class of languages as MATs (see Section 2.1).

2. Synchronous MATs have the same power as MATs (see Section 2.3).

3. Synchronous SCGs are able to generate all recursively enumerable languages (see Section 2.2).

4. RTs can generate any language that can be generated by some MAT, and they can accept any language that can be accepted by some k-PBCA (see Section 3.1).

5. Leftmost restricted RTs can only accept and generate context-free languages (see Section 3.2). Note that this is still an increase in accepting power compared to FAs.

6. RTs with appearance checking can both accept and generate all recursively enumerable languages (see Section 3.3).

We have also discussed application perspectives of the new models in translation of natural languages, using select case studies from Czech, English, and Japanese to illustrate (see Chapter 4). Besides natural language processing, the models can be useful in other translation and transformation tasks, such as programming language compilation.

5.1 Further Research Prospects

Further research prospects include the study of other theoretical properties of the proposed models, such as descriptional complexity. Although we have already shown which language classes our new models define, how efficiently they can do so remains an open problem. That is, we can investigate the effects of different limits placed on, for example, the number of nonterminal symbols in grammars, states in automata, or rules in both. In MATs, we can also limit the length of matrices, and similarly in SCGs, the length of scattered context rules (as sequences of context-free rules).

As we have done with RTs by introducing an appearance checking and a leftmost restriction, we can consider other variants of our models and investigate their properties. For example, we could restrict SSCGs by using propagating SCGs (which are known to be strictly weaker than SCGs with erasing rules). We can also introduce and study systems consisting of other well-known grammars and automata.

Extension to more than two components is possible as well. In such case, we could further investigate the relations to known grammar systems (see [10], [31], or [33]) and automata systems (see [7], [11], or [28]).

Finally, note that although our synchronous grammars and RTs represent different approaches and, consequently, are defined differently, there is a significant similarity in their basic principles. In essence, they are all systems in which the cooperation of components is achieved by synchronization of their rules. It might be useful to introduce a more general formalism allowing for various components, and thus encompassing all such rule-synchronized systems.

From a more practical viewpoint, an important area to investigate is syntax analysis. For practical applications, we need to be able to parse sentences efficiently. There are well-known parsing methods for CFGs, such as (generalized) LR parsing or chart parsing, but for models with regulated rewriting, the situation is more complicated. While there have been some research in this area, particularly for SCGs (see [24] or [38]), efficient parsing with matrix grammars and scattered context grammars still represents an open problem.

In the examples presented in this work, we have made two important assumptions. First, we already have the input sentence analysed on a low level—that is, we know where every word starts and ends (which may be a non-trivial problem in itself in some languages, such as Japanese) and have some basic grammatical information about it. Furthermore, we assume that we know the translation of the individual words.

For practical applications in natural language translation, we would need a more complex system, with at least two other components: a part-of-speech tagger (lexical analyzer), and a dictionary to translate the actual meanings of the words (although we can do this directly within a grammar by using dictionary rules, as shown in Example 4.2, a separate dictionary generally allows for more efficient encoding). Then, the component based on the discussed formal models could be used to transform the syntactic structure of a sentence and ensure that the words in the translated sentence are in the correct form.

Bibliography

[1] S. Abraham. Some questions of language theory. In *Proceedings of the 1965 conference on Computational linguistics*, COLING '65, pages 1–11, Stroudsburg, PA, USA, 1965. Association for Computational Linguistics.

[2] A. V. Aho. *Compilers: Principles, Techniques, and Tools*. Pearson/Addison Wesley, 2007.

[3] A. V. Aho and J. D. Ullman. Syntax directed translations and the pushdown assembler. *J. Comput. Syst. Sci.*, 3(1):37–56, February 1969.

[4] J. Allen. *Natural language understanding (2nd edition)*. Benjamin/Cummings series in computer science. Benjamin/Cummings Pub. Co., 1995.

[5] O. Bojar and M. Čmejrek. Mathematical model of tree transformations. In *Project Euromatrix Deliverable 3.2*, Prague, 2007. Charles University.

[6] M. Čermák, P. Horáček, and A. Meduna. Rule-restricted automaton-grammar tranduscers: Power and linguistic applications. *Mathematics for Applications*, 1(1):13–35, 2012.

[7] M. Čermák and A. Meduna. n-accepting restricted pushdown automata systems. In *13th International Conference on Automata and Formal Languages*, pages 168–183. Computer and Automation Research Institute, Hungarian Academy of Sciences, 2011.

[8] D. Chiang. An introduction to synchronous grammars. In *44th Annual Meeting of the Association for Computational Linguistics*, 2006.

[9] D. Chiang. *Grammars for Language and Genes: Theoretical and Empirical Investigations*. Theory and applications of natural language processing. Springer, 2011.

[10] E. Csuhaj-Varju, J. Kelemen, Gh. Paun, and J. Dassow, editors. *Grammar Systems: A Grammatical Approach to Distribution and Cooperation*. Gordon and Breach Science Publishers, Inc., Newark, NJ, USA, 1st edition, 1994.

[11] E. Csuhaj-Varjú, C. Martín-Vide, V. Mitrana, and Gy. Vaszil. Parallel communicating pushdown automata systems. *Int. J. Found. Comput. Sci.*, 11(4):633–650, 2000.

[12] J. Dassow and Gh. Păun. *Regulated Rewriting in Formal Language Theory*. Springer-Verlag, Berlin, 1989.

[13] S. Ginsburg. *Algebraic and Automata-Theoretic Properties of Formal Languages.* Elsevier Science Inc., New York, NY, USA, 1975.

[14] S. A. Greibach. Remarks on blind and partially blind one-way multicounter machines. *Theor. Comput. Sci.*, 7:311–324, 1978.

[15] E. M. Gurari and O. H. Ibarra. A note on finite-valued and finitely ambiguous transducers. *Theory of Computing Systems*, 16:61–66, 1983. 10.1007/BF01744569.

[16] J. Hajič. *Disambiguation of Rich Inflection: Computational Morphology of Czech.* Wisconsin Center for Pushkin Studies. Karolinum, 2004.

[17] J. E. Hopcroft, R. Motwani, and J. D. Ullman. *Introduction to Automata Theory, Languages, and Computation.* Addison Wesley, 2000.

[18] P. Horáček. Formal models in processing of japanese language. In *Proceedings of the 16th Conference and Competition STUDENT EEICT 2010 Volume 5*, pages 161–165. Brno University of Technology, 2010.

[19] P. Horáček. Parse driven translation. In *Proceedings of the 17th Conference and Competition STUDENT EEICT 2011 Volume 3*, pages 480–484. Brno University of Technology, 2011.

[20] P. Horáček. On generative power of synchronous grammars with linked rules. In *Proceedings of the 18th Conference STUDENT EEICT 2012 Volume 3*, pages 376–380. Brno University of Technology, 2012.

[21] P. Horáček. Application perspectives of synchronous matrix grammars. In *Proceedings of the 19th Conference STUDENT EEICT 2013 Volume 3*, pages 202–206. Brno University of Technology, 2013.

[22] P. Horáček and A. Meduna. Regulated rewriting in natural language translation. In *7th Doctoral Workshop on Mathematical and Engineering Methods in Computer Science*, pages 35–42, Brno, CZ, 2011. Brno University of Technology.

[23] P. Horáček and A. Meduna. Synchronous versions of regulated grammars: Generative power and linguistic applications. *Theoretical and Applied Informatics*, 24(3):175–190, 2012.

[24] O. Jirák and D. Kolář. Comparison of classical and lazy approach in scg compiler. In *NUMERICAL ANALYSIS AND APPLIED MATHEMATICS ICNAAM 2011: International Conference on Numerical Analysis and Applied Mathematics*, volume 1389, pages 873–876. American Institute of Physics, 2011.

[25] O. Jirák and Z. Křivka. Design and implementation of back-end for picoblaze c compiler. In *Proceedings of the IADIS International Conference Applied Computing 2009*, pages 135–138. International Association for Development of the Information Society, 2009.

[26] M. Khalilov and J. A. R. Fonollosa. N-gram-based statistical machine translation versus syntax augmented machine translation: comparison and system combination. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics*, EACL '09, pages 424–432, Stroudsburg, PA, USA, 2009. Association for Computational Linguistics.

[27] P. M. Lewis and R. E. Stearns. Syntax-directed transduction. *J. ACM*, 15(3):465–488, July 1968.

[28] C. Martín and V. Mitrana. Parallel communicating automata systems. *Journal of Applied Mathematics and Computing*, pages 237–257, 2008.

[29] A. Meduna. A trivial method of characterizing the family of recursively enumerable languages by scattered context grammars. *EATCS Bulletin*, 1995(56):1–3, 1995.

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

[31] A. Meduna and R. Lukáš. Multigenerative grammar systems. *Schedae Informaticae*, 2006(15):175–188, 2006.

[32] A. Meduna and J. Techet. *Scattered Context Grammars and their Applications.* WIT Press, UK, GB, 2010.

[33] R. Meersman and G. Rozenberg. Cooperating grammar systems. In J. Winkowski, editor, *Mathematical Foundations of Computer Science 1978*, volume 64 of *Lecture Notes in Computer Science*, pages 364–373. Springer Berlin Heidelberg, 1978.

[34] T. Mine, R. Taniguchi, and M. Amamiya. Coordinated morphological and syntactic analysis of japanese language. In *Proceedings of the 12th international joint conference on Artificial intelligence - Volume 2*, pages 1012–1017. Morgan Kaufmann Publishers Inc., 1991.

[35] R. Mitkov, editor. *The Oxford Handbook of Computational Linguistics.* Oxford University Press, 2003.

[36] M. Mohri. Finite-state transducers in language and speech processing. *Comput. Linguist.*, 23(2):269–311, June 1997.

[37] S. S. Muchnick. *Advanced compiler design and implementation.* Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1997.

[38] F. Popowich. Chart parsing of scattered context grammars. *Applied Mathematics Letters*, 7(1):35–40, 1994.

[39] G. Rozenberg and A. Salomaa, editors. *Handbook of Formal Languages.* Springer-Verlag, Berlin, 1997.

[40] P. Šaloun. Parallel LR parsing. In *Proceedings of the Fifth International Scientific Conference Electronic Computers and Informatics 2002.* The University of Technology Košice, 2002.

[41] A. Weber. On the valuedness of finite transducers. *Acta Informatica*, 27:749–780, 1990. 10.1007/BF00264285.

[42] A. Zollmann and A. Venugopal. Syntax augmented machine translation via chart parsing. In *Proceedings of the Workshop on Statistical Machine Translation*, StatMT '06, pages 138–141, Stroudsburg, PA, USA, 2006. Association for Computational Linguistics.

Appendix A

Curriculum Vitae

Personal Information

Name	Petr Horáček
Degree	Ing.
Citizenship	Czech Republic
Nationality	czech
Date of Birth	December 24, 1983
Birthplace	Nové Město na Moravě

Contact Information

Address	Beranka 1360, 593 01 Bystřice nad Pernštejnem, CZ
E-mail	horacekp@seznam.cz
Phone	+420 605 871 469
LinkedIn	`http://www.linkedin.com/in/horacekp`

Education

2009 — present	Faculty of Information Technology, Brno University of Technology • doctoral study (PhD.) • theoretical computer science, formal language theory, natural language processing • `http://www.fit.vutbr.cz/~ihoracekp/`
2007 — 2009	Faculty of Information Technology, Brno University of Technology • master study (Ing.) • specialization: Computer Graphics and Multimedia
2003 — 2007	Faculty of Information Technology, Brno University of Technology • bachelor study (Bc.) • specialization: Information Technology
1995 — 2003	Gymnázium Bystřice nad Pernštejnem