BRNO UNIVERSITY OF TECHNOLOGY
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

FACULTY OF INFORMATION TECHNOLOGY
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

DEPARTMENT OF INFORMATION SYSTEMS
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

ON PARALLEL PROCESSING IN FORMAL MODELS: JUMPING AUTOMATA AND NORMAL FORMS

O PARALELNÍM ZPRACOVÁNÍ VE FORMÁLNÍCH MODELECH

EXTENDED ABSTRACT OF A PHD THESIS
ROZŠÍŘENÝ ABSTRAKT DISERTAČNÍ PRÁCE

AUTHOR **Ing. RADIM KOCMAN**
AUTOR PRÁCE

SUPERVISOR **prof. RNDr. ALEXANDER MEDUNA, CSc.**
ŠKOLITEL

BRNO 2020

Abstract

The present thesis introduces and studies new possibilities of parallel processing in formal models. More specifically, it focuses its attention on parallel versions of jumping finite automata and on normal forms of grammars with interesting parallel properties.

In the first part of this thesis, we give an initial motivation for studying parallel processing in formal models. We briefly introduce jumping models and normal forms of grammars and grammar systems. Finally, we state the precise focus and goals of our research.

The second part of this thesis is focused on new results on jumping finite automata. First, we introduce n-parallel jumping finite automata that enhance the original jumping finite automaton model with multiple reading heads. The rest of the chapter then studies the accepting power of the model under two different jumping modes. Second, we introduce double-jumping finite automata and explore advanced jumping modes utilizing two heads. We study the accepting power of the models and also the closure properties of the related language families. Lastly, we introduce jumping $5' \to 3'$ Watson-Crick finite automata that combine the jumping behavior with the biology-inspired Watson-Crick finite automata that process double-stranded DNA sequences. The rest of this chapter then studies the accepting power of the model under unrestricted and various restricted conditions.

The third part of this thesis is focused on new results on CD grammar systems. We introduce two types of transformations that turn arbitrary general grammars into equivalent two-component general CD grammar systems of very reduced and simplified forms. Apart from the reduction and simplification, we describe several other useful properties concerning these systems and the way they work.

In the last part, we mention possible application perspectives for the introduced models and normal forms, and we conclude the thesis with the final summary and the description of theoretical perspectives for the achieved results.

Keywords

parallel processing, discontinuous tape reading, parallel tape reading, normal forms, simulated non-context-free rules, homogeneous rules, evenly homogeneous rules, general grammars, jumping finite automata, left and right jumps, n-parallel right linear grammars, even-length languages, Watson-Crick finite automata, CD grammar systems

Reference

KOCMAN, Radim. *On Parallel Processing in Formal Models: Jumping Automata and Normal Forms*. Brno, 2020. PhD thesis. Brno University of Technology, Faculty of Information Technology. Supervisor prof. RNDr. Alexander Meduna, CSc.

Contents

Chapter 1

Introduction

In this chapter, we give an initial motivation for studying parallel processing in formal models. We briefly introduce jumping models—quite a new group of formal models focused on discontinuous information processing, which were not yet studied together with parallel mechanisms—and normal forms of grammars and grammar systems—the common unifying forms of definitions of formal models, which are usually not concerned with parallelism. Afterwards, we state our focus and goals for studying parallelism together with jumping automata and normal forms. This thesis assumes that the reader is firmly familiar with the basic notions from the theory of automata and formal languages, and thus we use them here extensively without further explanation. The more advanced terminology is introduced in greater detail in the full thesis.

1.1 Parallelism

When we talk about parallelism in modern computer science, we almost automatically mean some form of parallel processing or parallel computing. By these terms we refer to situations where we want to split some large task into smaller chunks of work in such a way that the chunks can be executed in parallel on separate processing units, and the whole task can thus be computed faster than if it was executed completely sequentially on a single processing unit.

Nonetheless, this perception of the notion of parallelism can change quite rapidly when we wander into more theoretical branches of computer science; especially if we consider the basic research in the theory of formal languages. There are many formal models in this area that incorporate some form of parallelism, but they utilize very diverse mechanics in the background to achieve their goal. If we take a broader look at these formal models and the basic research in general, we can roughly divide parallelism in formal language theory into the following three categories:

(P.1) parallelism that increases the expressive power of the model,

(P.2) parallelism that is a fundamental part of the behavior of the model,

(P.3) parallelism that splits the work of the task.

Parallelism That Increases the Expressive Power of the Model

The most commonly studied category in the basic research is probably category (P.1). This is especially noticeable in formal grammars. Consider classical formal grammars in general, there is a big difference if a model can use only context-free rules or also non-context-free rules. It is much harder to deal with the non-context-free rules from both the theoretical and practical point of view. Therefore, there is a large incentive to study models that can use only the context-free (or even more restricted) rules but that also incorporate some additional mechanisms which further increase their generative power.

In formal grammars, the models can incorporate parallelism in such a way that, in each step of the rewriting process, the grammar rewrites several symbols in the sentential form at once in parallel. Let us mention some well-known models that match this description:

- scattered context grammars (see [21, 58, 54]),
- simple matrix grammars (see [25, 74, 89, 90]),
- equal matrix grammars (see [80]),
- n-parallel (right-)linear grammars (see [75, 89, 90, 88, 73, 74]).

In the case of finite automata, we can imagine the parallelism of category (P.1) as a parallel cooperation of multiple heads. There are several well-known models of finite automata that utilize more than one head, nonetheless, their behavior do not fall precisely into one specific category of parallelism; so we will leave their description for later.

A very common property of models from this category is that we can freely select their degree of parallelism. More specifically, we can choose n which represents the number of symbols or heads that are considered together in a single step of the model. Then, if $n = 1$, we get the power of a classical non-augmented model (e.g., the power of context-free grammars); and, for $n > 1$, we either get an infinite hierarchy of more powerful models or the power of the model increases at first but then stabilizes. Due to this common property, we can also include parallel communicating (PC) grammar systems (see [8, 78]) into this category since they behave very similarly in this regard.

Parallelism That Is a Fundamental Part of the Behavior of the Model

Considering category (P.2), we are looking at the models that have parallelism rooted inseparably into their core structure. From our exploration of this topic, it seems that the models which fall into this category are usually related to biology.

On the one hand, there are massively parallel models such as Lindenmayer systems (see [77, 58, 54]) that are based on the evolution process. In these models, all eligible symbols in the sentential form are always rewritten together at once in parallel. Consequently, it is not possible to select a constant degree of parallelism for these models since the conditions continuously change depending on the current task.

On the other hand, there are also models with a fixed degree of parallelism such as Watson-Crick finite automata (see [72]). These automaton models use two heads in parallel in such a way that each head processes one strand of a double-stranded DNA input sequence. Consequently, the degree of parallelism of Watson-Crick finite automata is always two.

Parallelism That Splits the Work of the Task

Lastly, if we consider category (P.3) in the basic research, it seems that there is not much interest to study possibilities how to split the work for the given tasks. This may not be that surprising because in the basic research we usually study characteristics like the expressive power, closure properties, and the decidability and complexity of various operations; and, of course, these results are not affected by parallelism. We often even prefer approaches that are completely sequential because it makes the subsequent proof techniques much easier in many cases. When we do consider parallelism that splits the work of the tasks (see [77, 78]), we usually just simply conclude that if the model behaves nondeterministically, then we can explore different cases in parallel, and if the model uses only context-free rules, then we can trivially split the generation process into multiple independent parts.

It is possible to find some theoretical papers that explore this role of parallelism further in certain areas, e.g., in biomolecular computing (see [43]); but a thorough study is usually left for practical applications such as parsing (see [22]), formal verification, and others.

Parallelism and Finite Automata

The situation around the types of parallelism gets more complex if we look at finite automata. Thus, we introduce some additional categorization.

There are some finite automaton models that have the same expressive power as grammars from category (P.1). For example, self-regulating finite automata (see [52]), pure multi-pushdown automata that perform complete pushdown pops (see [48]), and finite-turn checking automata (see [81]), which are connected to the various versions of simple matrix, equal matrix, and n-parallel right-linear grammars. However, we do not consider these models to be parallel. This is due to the fact that, up until quite recently, automaton models always read the input tape almost exclusively in the strictly continuous (left-to-right) symbol-by-symbol way. The mentioned models are no exceptions, and thus they use various kinds of stacks to match the expressive power of the parallel grammars but otherwise work strictly continuously on the input tape in a completely non-parallel way.

As we have already pointed out, we can imagine parallelism in finite automata as a parallel cooperation of multiple heads. There is indeed the well-known concept of Turing machines with multiple tapes and multiple heads; which was also adapted and studied in terms of finite automaton models. Nonetheless, not all such models necessarily work in a parallel way. Considering multi-head finite automata that actually do work in a parallel way, we can find two distinct categories of their behavior:

(PA.1) multi-head automata where each head works on an independent copy of the input,

(PA.2) multi-head automata where heads cooperate to process the single input.

The first category seems to be the most studied one so far. Let us mention some prominent models that fit into this description: classical Watson-Crick finite automata (see [72]), multi-head finite automata (see [76, 28, 83, 24]), and parallel communicating finite automaton systems (see [24]). In these models, the heads can work in parallel, however, their behavior can be hardly seen as parallel processing since it does not speed up the task in any way. In most cases, there is a single read-only input tape that must be completely traversed with all heads until the conclusion about the acceptance of the input is reached.

We only know about a few models that fall into the second category. These are finite automaton models introduced by Nagy that utilize two heads with the following behavior.

The first head reads the input from left to right, the second head reads the input from right to left, and the processing of the input ends when the heads meet each other on the tape. This concept was explored several times in various models: 2-head finite automata (see [64]), $5' \to 3'$ Watson-Crick finite automata (see [60, 61, 62, 63, 65, 70, 71, 69]), multicounter $5' \to 3'$ Watson-Crick finite automata (see [11, 59, 23]), and two-head finite-state acceptors with translucent letters (see [67, 68]). In these models, the heads truly cooperate in parallel on a single tape; thus, this behavior can be seen as parallel processing. Naturally, their degree of parallelism is always two.

1.2 Jumping Models

The idea of a jumping mechanism that is integrated deeply into the core behavior of formal models is quite a new concept that was first proposed in 2012 by Meduna and Zemek in [57]. The main motivation behind this concept is the fact that in the previous century most classical computer science methods were developed for continuous information processing, but in modern computation methods we often process information in a discontinuous way. The continuous processing approach is deeply rooted in classical formal models such as finite automata which traditionally process the input information in a strictly continuous left-to-right symbol-by-symbol way. Therefore, it makes sense to introduce and study jumping mechanisms that can more appropriately represent the behavior of modern computation methods that often have to jump over large portions of the input information between individual steps of the process.

In the following years, this idea got a lot of traction among other researchers in the field of formal language theory. At the time of writing, there are around 30 papers that study jumping models in various ways, and this number is still increasing. From the theoretical point of view, these models have an interesting characteristic that, on the one hand, they often define language families that are outside the usual Chomsky hierarchy, but, on the other hand, they are still related to some other well-known mathematical models. With this characteristic, it is possible to combine results from different fields that previously looked unrelated. It is out of the scope of this thesis to cover all studied models, but we at least give a brief overview of the most influential ones.

Jumping Finite Automata

The definition of a jumping finite automaton was first introduced by Meduna and Zemek in [57], and it can be also found in the follow-up books [58, 54].

Let us first recall the notion of a classical finite automaton, M, which consists of an input tape, a reading head, and a finite state control. The input tape is divided into squares. Each square contains one symbol of an input string. The symbol under the reading head, a, is the current input symbol. The finite control is represented by a finite set of states together with a control relation, which is usually specified as a set of computational rules. The automaton M computes by making a sequence of moves. Each move is made according to a computational rule that describes how the current state is changed and whether the current input symbol is read. If the symbol is read, the reading head is shifted precisely one square to the right. M has one state defined as the start state and some states designated as final states. If M can read w by making a sequence of moves from the start state to a final state, M accepts w; otherwise, M rejects w.

In essence, a jumping finite automaton works just like a classical finite automaton except it does not read the input tape in a symbol-by-symbol left-to-right way. After the automaton reads a symbol, the head can jump over (skip) a portion of the tape in either direction. Once an occurrence of a symbol is read on the tape, it cannot be re-read again later. Otherwise, it coincides with the standard notion of a finite automaton.

Apart from the definition, the paper [57] studies the accepting power, decidability properties, and closure properties of the model under various restrictions. Surprisingly, compared to classical finite automata, there is a significant difference if the model is a general jumping finite automaton (GJFA), which can read multiple symbols in a step, or a non-general jumping finite automaton (JFA), which can read only a single symbol in a step.

Concerning GJFAs, there are papers written by Vorel (see [84, 85, 86]) that continue the investigation of decidability and closure properties. Moreover, they connect GJFAs with graph-controlled insertion systems and Galiukschov semicontextual grammars.

Concerning both GJFAs and JFAs, there are papers written by Fernau, Paramasivan, Schmid, and Vorel (see [14, 15]) that present a large number of various new results and also connect JFAs with shuffle languages, commutative context-free grammars, letter bounded languages, and regular expressions over comutative monoids.

Lastly, concerning JFAs, there are papers written by Beier, Holzer, and Kutrib (see [4, 5]) that study their operational state complexity and decidability and also connect JFAs with semilinear sets and Parikh images of regular sets.

Other Jumping Models

The other two influential models are covered in detail in the full thesis:

- jumping grammars (see [31, 54, 44, 45]),
- one-way jumping finite automata (see [6, 7, 13, 2, 1, 3, 12]).

Besides the most influential models mentioned previously, there are also other papers that study the jumping mechanism further in more advanced formal models:

- two-dimensional jumping finite automata (see [26, 46, 27]),
- jumping scattered context grammars (see [53, 54]),
- jumping pure grammars (see [29]),
- jumping restarting automata (see [87]),
- jumping multi-head automata (see [41]),
- Watson-Crick jumping finite automata (see [47]).

Note that it may seem, from the names of jumping multi-head automata and Watson-Crick jumping finite automata, that these models are similar to the models studied later in this thesis. However, both of the mentioned models fall into the category (PA.1) of parallelism in finite automata. On the other hand, all finite automata studied in this thesis fall into the category (PA.2) which is a fundamentally different behavior.

1.3 Normal Forms and Grammar Systems

Moving away from the idea of the jumping mechanism, we need to introduce the remaining two concepts that are also studied together with parallelism in this thesis.

Normal Forms

A classical (general) grammar G contains production rules of the form $x \to y$, where x and y are strings over the alphabet of G. If we want to change the generative power of the grammar, we can put restrictions on the form of the rules. Classically, we consider some types of monotonous, context-sensitive, context-free, ε-free, linear, right-linear, and regular restrictions (see, e.g., [77, 31]). Nonetheless, even in these cases, the forms of rules are often still rather loose. This can be an unwanted property from both the theoretical and practical point of view because the follow-up proofs and algorithms have to take into account all possible forms of the definition of the grammar. Therefore, there is an incentive in formal language theory to study normal forms of grammars and grammar systems that severely restrict the possible forms of the definition of the model but, in the same time, keep the generative power intact.

We skip the description of basic normal forms that handle only grammars with context-free rules since, in these cases, it is rather easy to work with them in a parallel way. However, let us consider some well-known normal forms for general grammars: Kuroda normal form, Penttonen normal form, and Geffert normal forms (see, e.g., [58]). These normal forms are all frequently used in formal language theory. However, we argue that none of them has particularly fitting parallel properties. When we want to construct a parallel rewriting process for general grammars, the non-context-free rules really complicate the task since there is no simple way how to split the generation of a sentence into multiple independent parts, and the classical normal forms do not help with this matter.

First, consider an unrestricted general grammar. There can be a large number of non-context-free rules. These rules can work with very large contexts since there is no bound on the length of x in $x \to y$. Furthermore, the non-context-free rules can be also used anywhere in the generation process.

Second, consider Kuroda and Penttonen normal forms. Indeed, the required context of the non-context-free rules is now minimal. However, there can still be a large number of these rules, and they can still be used anywhere in the generation process.

Lastly, consider Geffert normal forms. There is a limited number of non-context-free rules, and they work with small contexts. Nonetheless, all Geffert normal forms share the same deeply-rooted property that, in any generated string, there is always at most one position that can be rewritten with the rules of the grammar. In some situations, this property can be highly valuable from both the theoretical and practical point of view; however, this complicates the construction of a parallel rewriting process even further.

Besides the classical normal forms, we can find many other normal forms for various formal models (see, e.g., [58, 54, 51]). However, it seems that in almost all cases the definitions are primarily focused only on the very restricted forms of rules, minimum number of non-context-free rules, and minimum number of nonterminals. Consequently, they do not care about their resulting parallel properties.

CD Grammar Systems

In essence, a cooperating distributed (CD) grammar system (see [8]) can be seen as an extension of a classical grammar. It has not one but multiple finite sets of production rules (components), and the rewriting process can operate in various complex modes that control which sets of production rules can be currently used. From another perspective, a CD grammar system can be seen as a group of grammars that distribute their work and cooperate on a single string to produce the final sentence.

Considering the core behavior of CD grammar systems, their extension over general grammars is not parallel in nature because the modes switch between components in a strictly sequential way. However, we find this model useful for the study of parallel properties of normal forms. To give a brief insight, with CD grammar systems, we can strictly split the context-free and non-context-free rules into different components, and we can clearly divide the rewriting process into several phases that use different types of rules. This can help us to pinpoint opportunities for a viable parallel rewriting process.

1.4 Specification of Goals

The principal focus of this thesis is the theoretical study of parallelism in the areas of formal language theory where this approach was not yet thoroughly considered. First, we explore parallel processing with jumping finite automata. Second, we introduce new normal forms designed for parallel rewriting.

New Results on Jumping Automata

The first unexplored area can be easily seen if we take a closer look at the previous description of parallelism in classical finite automata and the new jumping mechanism introduced in jumping finite automata. Once we shift our attention to discontinuous information processing, and we are no longer restricted with the classical reading in a continuous left-to-right symbol-by-symbol way, there are a lot of new opportunities how to design the behavior of multi-head finite automaton models. From the point of view of parallelism in finite automata, we want to focus on category (PA.2) where the heads work in parallel to process the single input. From the point of view of general parallelism, we want to design models that fit into category (P.3) but also share some similarities with categories (P.1) and (P.2). To be more precise, we will introduce and study new parallel versions of jumping finite automata. Since these mechanisms were not yet studied together, this research should lead to some novel results that are usually not observed in classical models. Moreover, we should be able to find some new close connections with different formal models.

New Results on CD Grammar Systems

The second unexplored area was already foreshadowed in the description of normal forms of grammars and grammar systems. We want to introduce new normal forms that are focused not just on the usual restrictive properties but also on the resulting parallel properties. More precisely, we will use an extended version of CD grammar systems that can accept recursively-enumerable languages, and we will introduce new normal forms for these grammar systems that will have a very limited number of non-context-free rules and that will be suitable for a parallel rewriting process. Such normal forms can be interesting from both the theoretical and practical point of view. This thesis is focused primarily on the theoretical aspects of the topic, but we will also mention some further ideas in the conclusion.

Chapter 2

n-Parallel Jumping Finite Automata

This chapter covers our first steps to explore the possibilities of parallel jumping finite automata. The content of this chapter is composed of results that were presented at the conferences Excel@FIT 2015 (see [32]) and SDOT 2015 (see [37]) and also a few additional unpublished results. We define a modification of jumping finite automata which read input words discontinuously with multiple synchronized heads. Moreover, we also define a more restricted mode for these automata which uses only the right jumps.

Definition. For $n \geq 1$, an *n-parallel general jumping finite automaton*, an *n-PGJFA* for short, is a quintuple

$$M = (Q, \Sigma, R, S, F),$$

where Q is a finite set of states, Σ is an input alphabet, $Q \cap \Sigma = \emptyset$, $R \subseteq Q \times \Sigma^* \times Q$ is finite, $S \subseteq Q^n$ is a set of start state strings, and $F \subseteq Q$ is a set of final states. Members of R are referred to as rules of M and instead of $(p, y, q) \in R$, we write $py \to q \in R$.

A *configuration* of M is any string in $\Sigma^* Q \Sigma^*$. Let X denote the set of all configurations over M. The binary jumping relation, symbolically denoted by $\curvearrowright$, over X, is defined as follows. Let $x, z, x', z' \in \Sigma^*$ such that $xz = x'z'$ and $py \to q \in R$; then, M makes a *jump* from $xpyz$ to $x'qz'$, symbolically written as

$$xpyz \curvearrowright x'qz'.$$

Let \$ be a special symbol, $\$ \notin (Q \cup \Sigma)$. An *$n$-configuration* of M is any string in $(X\{\$\})^n$. Let ${}_nX$ denote the set of all n-configurations over M. The binary n-jumping relation, symbolically denoted by ${}_n\curvearrowright$, over ${}_nX$, is defined as follows. Let $\zeta_1\$ \cdots \zeta_n\$, \vartheta_1\$ \cdots \vartheta_n\$ \in {}_nX$, so $\zeta_i, \vartheta_i \in X$, $1 \leq i \leq n$; then, M makes an *n-jump* from $\zeta_1\$ \cdots \zeta_n\$$ to $\vartheta_1\$ \cdots \vartheta_n\$$, symbolically written as

$$\zeta_1\$ \cdots \zeta_n\$ \ {}_n\curvearrowright \ \vartheta_1\$ \cdots \vartheta_n\$$$

if and only if $\zeta_i \curvearrowright \vartheta_i$ for all $1 \leq i \leq n$. In the standard manner, we extend ${}_n\curvearrowright$ to ${}_n\curvearrowright^m$, where $m \geq 0$. Let ${}_n\curvearrowright^+$ and ${}_n\curvearrowright^*$ denote the transitive closure of ${}_n\curvearrowright$ and transitive-reflexive closure of ${}_n\curvearrowright$, respectively.

The language accepted by M, denoted by $L(M, n)$, is defined as

$$L(M, n) = \{u_1 v_1 \cdots u_n v_n : u_1 s_1 v_1\$ \cdots u_n s_n v_n\$ \ {}_n\curvearrowright^* \ f_1\$ \cdots f_n\$, \\ u_i, v_i \in \Sigma^*, \ s_1 \cdots s_n \in S, \ f_i \in F, \ 1 \leq i \leq n\}.$$

Let $w \in \Sigma^*$. We say that M accepts w if and only if $w \in L(M, n)$. M rejects w if and only if $w \in \Sigma^* - L(M, n)$.

Definition. For $n \geq 1$, let $M = (Q, \Sigma, R, S, F)$ be an n-PGJFA, and let X denote the set of all configurations over M. The binary right jumping relation, symbolically denoted by ${}_r\curvearrowright$, over X, is defined as follows. Let $w, x, y, z \in \Sigma^*$, and $py \to q \in R$; then, M makes a *right jump* from $wpyxz$ to $wxqz$, symbolically written as

$$wpyxz \ {}_r\curvearrowright \ wxqz.$$

Let ${}_nX$ denote the set of all n-configurations over M. The binary right n-jumping relation, symbolically denoted by ${}_{n-r}\curvearrowright$, over ${}_nX$, is defined as follows. Let $\zeta_1\$ \cdots \zeta_n\$, \vartheta_1\$ \cdots \vartheta_n\$ \in {}_nX$, so $\zeta_i, \vartheta_i \in X$, $1 \leq i \leq n$; then, M makes a *right n-jump* from $\zeta_1\$ \cdots \zeta_n\$$ to $\vartheta_1\$ \cdots \vartheta_n\$$, symbolically written as

$$\zeta_1\$ \cdots \zeta_n\$ \ {}_{n-r}\curvearrowright \ \vartheta_1\$ \cdots \vartheta_n\$$$

if and only if $\zeta_i \ {}_r\curvearrowright \vartheta_i$ for all $1 \leq i \leq n$.

We extend ${}_{n-r}\curvearrowright$ to ${}_{n-r}\curvearrowright^m$, ${}_{n-r}\curvearrowright^+$, and ${}_{n-r}\curvearrowright^*$, where $m \geq 0$, by analogy with extending the corresponding notations for ${}_n\curvearrowright$. Let $L(M, n{-}r)$ denote the language accepted by M using only right n-jumps.

Let **GJFA**, ${}_r$**GJFA**, n-**PGJFA**, ${}_rn$-**PGJFA**, and n-**PRLG** denote the language families accepted by general jumping finite automata, general jumping finite automata using only right jumps, n-PGJFAs, n-PGJFAs using only right n-jumps, and n-parallel right linear grammars, respectively.

Most notably, this chapter shows that both modifications extend their original models and that they define infinite hierarchies of language families based on the number of reading heads. In other words, every additional head always increases the power of the model.

Theorem. 1-***PGJFA*** $=$ ***GJFA***.

Theorem. *For all* $n \geq 1$, n-***PGJFA*** $\subset$ $(n{+}1)$-***PGJFA***.

Theorem. *For all* $n \geq 1$, n-***PGJFA*** $\subset$ ***CS***.

Theorem. ${}_r1$-***PGJFA*** $=$ ${}_r$***GJFA*** $=$ ***REG***.

Theorem. *For all* $n \geq 1$, ${}_rn$-***PGJFA*** $\subset$ ${}_r(n{+}1)$-***PGJFA***.

Theorem. *For all* $n \geq 1$, ${}_rn$-***PGJFA*** $\subset$ ***CS***.

Moreover, we show that the restricted mode creates a direct counterpart to n-parallel right linear grammars.

Lemma. *For every* n-PRLG $G = (N_1, \ldots, N_n, T, S1, P)$, *there is an* n-PGJFA $M = (Q, \Sigma, R, S2, F)$ *using only right* n*-jumps such that* $L(M, n{-}r) = L(G)$.

Lemma. *For every* n-PGJFA $M = (Q, \Sigma, R, S2, F)$ *using only right* n*-jumps, there is an* n-PRLG $G = (N_1, \ldots, N_n, T, S1, P)$ *such that* $L(G) = L(M, n{-}r)$.

Theorem. ${}_rn$-***PGJFA*** $=$ n-***PRLG***.

Chapter 3

Double-Jumping Finite Automata

This chapter studies the advanced possibilities of the two-head jumping finite automaton model under various reading modes. The content of this chapter is composed of results that were published at the conference NCMA 2016 (see [33]) and in the journal RAIRO (see [34]). Consider the notion of a jumping finite automaton M. We modify the way M works so it simultaneously performs two jumps according to the same rule. For either of the two jumps, it always considers three natural directions—(1) to the left, (2) to the right, and (3) in either direction.

Definition. Let $M = (Q, \Sigma, R, s, F)$ be a GJFA. Let $w, x, y, z \in \Sigma^*$ and $h : (p, y, q) \in R$; then, $wpyxz \underset{\blacktriangleright}{\curvearrowright} wxqz$ $[h]$ and $wxypz \underset{\blacktriangleleft}{\curvearrowright} wqxz$ $[h]$ in M.

Let X denote the set of all configurations of M. A 2-*configuration* of M is any string in XX. Let X^2 denote the set of all 2-configurations of M. For brevity, let $t_1t_2 \in \{\blacklozenge\blacklozenge, \blacktriangleright\blacktriangleright, \blacktriangleright\blacktriangleleft, \blacktriangleleft\blacktriangleright, \blacktriangleleft\blacktriangleleft\}$ such that $t_1, t_2 \in \{\blacklozenge, \blacktriangleright, \blacktriangleleft\}$. The binary t_1t_2 2-*jumping relation*, symbolically denoted by $\underset{t_1t_2}{\curvearrowright}$, over X^2, is defined as follows. Let $\zeta_1\zeta_2, \vartheta_1\vartheta_2 \in X^2$, where $\zeta_1, \zeta_2, \vartheta_1, \vartheta_2 \in X$, and $h \in R$; then, M makes a t_1t_2 2-*jump* from $\zeta_1\zeta_2$ to $\vartheta_1\vartheta_2$ according to h, symbolically written as

$$\zeta_1\zeta_2 \ \underset{t_1t_2}{\curvearrowright} \ \vartheta_1\vartheta_2 \ \ [h]$$

if and only if $\zeta_1 \underset{t_1}{\curvearrowright} \vartheta_1$ $[h]$ and $\zeta_2 \underset{t_2}{\curvearrowright} \vartheta_2$ $[h]$. Depending on the specific type of jumps $\blacklozenge\blacklozenge$, $\blacktriangleright\blacktriangleright$, $\blacktriangleright\blacktriangleleft$, $\blacktriangleleft\blacktriangleright$, $\blacktriangleleft\blacktriangleleft$, we use the following naming: unrestricted, right-right, right-left, left-right, left-left 2-jumping relation (or 2-jump), respectively.

Let o be any of the jumping direct relations introduced above. In the standard way, we extend o to o^m, $m \geq 0$; o^+; and o^*. To express that M only performs jumps according to o, write M_o. If o is one of the relations $\underset{\blacklozenge}{\curvearrowright}$, $\underset{\blacktriangleright}{\curvearrowright}$, $\underset{\blacktriangleleft}{\curvearrowright}$, set

$$L(M_o) = \{uv : u, v \in \Sigma^*, \ usv \ \ o^* \ f, \ f \in F\}.$$

If o is one of the relations $\underset{\blacklozenge\blacklozenge}{\curvearrowright}$, $\underset{\blacktriangleright\blacktriangleright}{\curvearrowright}$, $\underset{\blacktriangleright\blacktriangleleft}{\curvearrowright}$, $\underset{\blacktriangleleft\blacktriangleright}{\curvearrowright}$, $\underset{\blacktriangleleft\blacktriangleleft}{\curvearrowright}$, set

$$L(M_o) = \{uvw : u, v, w \in \Sigma^*, \ usvsw \ \ o^* \ ff, \ f \in F\}.$$

$L(M_o)$ is referred to as the *language of* M_o. Set $\mathscr{L}_o = \{L(M_o) : M \text{ is a GJFA}\}$; $\mathscr{L}_o$ is referred to as the *language family accepted by GJFAs according to* o.

To illustrate this terminology, take $o = \underset{\blacklozenge\blacklozenge}{\curvearrowright}$. Consider $M_{\underset{\blacklozenge\blacklozenge}{\curvearrowright}}$. Notice that

$$L(M_{\underset{\blacklozenge\blacklozenge}{\curvearrowright}}) = \{uvw : u, v, w \in \Sigma^*, \ usvsw \ \ \underset{\blacklozenge\blacklozenge}{\curvearrowright}{}^* \ ff, \ f \in F\}.$$

$L(M_{\underset{\blacklozenge\blacklozenge}{\curvearrowright}})$ is referred to as the *language of* $M_{\underset{\blacklozenge\blacklozenge}{\curvearrowright}}$. Set $\mathscr{L}_{\underset{\blacklozenge\blacklozenge}{\curvearrowright}} = \{L(M_{\underset{\blacklozenge\blacklozenge}{\curvearrowright}}) : M \text{ is a GJFA}\}$; $\mathscr{L}_{\underset{\blacklozenge\blacklozenge}{\curvearrowright}}$ is referred to as the *language family accepted by GJFAs according to* $\underset{\blacklozenge\blacklozenge}{\curvearrowright}$.

Lastly, we define an auxiliary subfamily of the family of regular languages that is useful to the study of the accepting power of GJFAs that perform right-left and left-right 2-jumps.

Definition. Let $L_{m,n}$ be a *simply-expandable language* (SEL) over an alphabet Σ if it can be written as follows. Let m and n be positive integers; then,

$$L_{m,n} = \bigcup_{h=1}^{m} \left\{ u_{h,1}u_{h,2}\cdots u_{h,n}v_h^i v_h^i u_{h,n}\cdots u_{h,2}u_{h,1} : i \geq 0,\ u_{h,k}, v_h \in \Sigma^*,\ 1 \leq k \leq n \right\}.$$

Let **SEL** denote the family of SELs. The most notable results of this chapter are summarized in Figures 3.1 and 3.2.

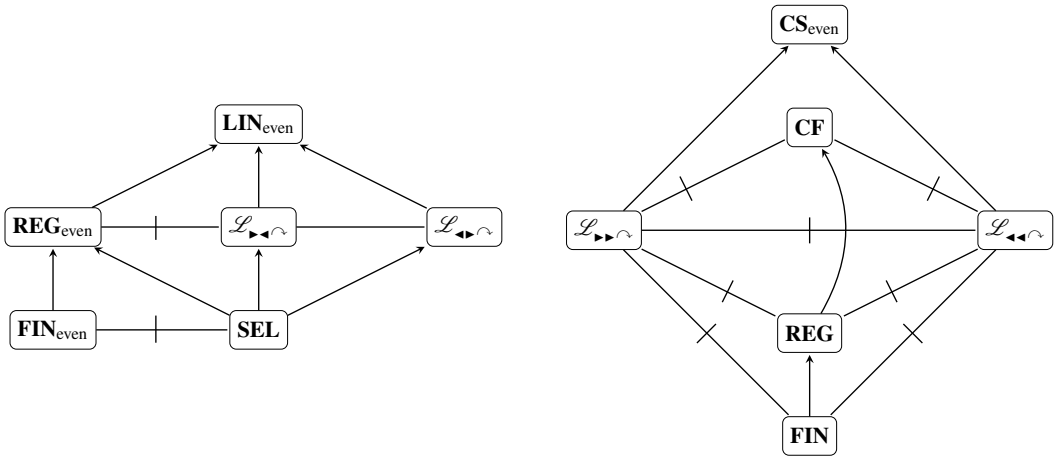

Figure 3.1: The hierarchies of closely related language families are shown. If there is a line or an arrow from family X to family Y in the figure, then $X = Y$ or $X \subset Y$, respectively. A crossed line represents the incomparability between connected families.

	$\mathscr{L}_{\blacktriangleright\blacktriangleleft\curvearrowright}$, $\mathscr{L}_{\blacktriangleleft\blacktriangleright\curvearrowright}$	$\mathscr{L}_{\blacktriangleright\blacktriangleright\curvearrowright}$	$\mathscr{L}_{\blacktriangleleft\blacktriangleleft\curvearrowright}$
endmarking (both sides)	− (+)	− (−)	− (−)
concatenation	−	−	−
square (L^2)	−	−	−
shuffle	−	−	−
union	+	+	+
complement	−	−	−
intersection	+	−	−
int. with regular languages	+	−	−
mirror image	+	−	−
finite substitution	−	−	−
homomorphism	+	−	−
ε-free homomorphism	+	−	−
inverse homomorphism	−	−	−

Figure 3.2: Summary of closure properties.

Chapter 4

Jumping $5' \to 3'$ Watson-Crick Finite Automata

This chapter studies a combined model of two-head jumping finite automata and sensing $5' \to 3'$ Watson-Crick finite automata. The content of this chapter is composed of results that were published at the conference NCMA 2018 (see [38]) and that are currently submitted to the journal Acta Informatica (see [36]).

Definition. A *jumping* $5' \to 3'$ *WK automaton* is a quintuple $M = (V, Q, q_0, F, \delta)$, where V, Q, q_0, and F are the same as in WK automata, $V \cap \{\#\} = \emptyset$, $\delta\colon (Q \times V^* \times V^* \times D) \to 2^Q$, where $D = \{\oplus, \ominus\}$ indicates the mutual position of heads, and $\delta(q, w_1, w_2, s) \neq \emptyset$ only for finitely many quadruples $(q, w_1, w_2, s) \in Q \times V^* \times V^* \times D$. We denote the head as $\blacktriangleright$-head or $\blacktriangleleft$-head if it reads from left to right or from right to left, respectively. We use the symbol $\oplus$ if the $\blacktriangleright$-head is on the input tape positioned before the $\blacktriangleleft$-head; otherwise, we use the symbol $\ominus$. A configuration (q, s, w_1, w_2, w_3) consists of the state $q \in Q$, the mutual position of heads $s \in D$, and the three unprocessed portions of the input tape: (a) before the first head (w_1), (b) between the heads (w_2), and (c) after the second head (w_3). A step of the automaton can be of the following four types: Let $x, y, u, v, w_2 \in V^*$ and $w_1, w_3 \in (V \cup \{\#\})^*$.

(1) $\oplus$-reading: $(q, \oplus, w_1, xw_2y, w_3) \curvearrowright (q', s, w_1\{\#\}^{|x|}, w_2, \{\#\}^{|y|}w_3)$, where $q' \in \delta(q, x, y, \oplus)$, and s is either $\oplus$ if $|w_2| > 0$ or $\ominus$ in other cases.
(2) $\ominus$-reading: $(q, \ominus, w_1y, \varepsilon, xw_3) \curvearrowright (q', \ominus, w_1, \varepsilon, w_3)$, where $q' \in \delta(q, x, y, \ominus)$.
(3) $\oplus$-jumping: $(q, \oplus, w_1, uw_2v, w_3) \curvearrowright (q, s, w_1u, w_2, vw_3)$, where s is either $\oplus$ if $|w_2| > 0$ or $\ominus$ in other cases.
(4) $\ominus$-jumping: $(q, \ominus, w_1\{\#\}^*, \varepsilon, \{\#\}^*w_3) \curvearrowright (q, \ominus, w_1, \varepsilon, w_3)$.

In the standard manner, let us extend $\curvearrowright$ to $\curvearrowright^n$, where $n \geq 0$; then, based on $\curvearrowright^n$, let us define $\curvearrowright^+$ and $\curvearrowright^*$. The accepted language, denoted by $L(M)$, can be defined by the final accepting configurations that can be reached from the initial one: A string w is accepted by a jumping $5' \to 3'$ WK automaton M if and only if $(q_0, \oplus, \varepsilon, w, \varepsilon) \curvearrowright^* (q_f, \ominus, \varepsilon, \varepsilon, \varepsilon)$, for $q_f \in F$.

Let **SWK** and **JWK** denote the language families accepted by sensing $5' \to 3'$ WK automata and jumping $5' \to 3'$ WK automata, respectively. Moreover, we are using prefixes **N**, **F**, **S**, **1**, **NS**, **FS**, **N1**, and **F1** to specify the restricted variants of jumping $5' \to 3'$ WK automata and appropriate language families. Finally, let $\mathbf{FIN}_{\varepsilon\text{-inc}}$ denote the family of finite languages that always contain the empty string.

This chapter shows that the new model is more powerful than sensing $5' \to 3'$ WK automata and some double-jumping reading modes. However, the resulting language family is incomparable with the language families accepted by (general) jumping finite automata.

Theorem. ***LIN*** $=$ ***SWK*** $\subset$ ***JWK***.

Proposition. *The language family accepted by double-jumping finite automata that perform right-left and left-right jumps is strictly included in* ***JWK***.

Proposition. ***JWK*** *is incomparable with* ***GJFA*** *and* ***JFA***.

To formally prove our results, we also introduce a new proof technique that utilizes the concept of the debt of a configuration in jumping $5' \to 3'$ WK automata.

Definition. Let $M = (V, Q, q_0, F, \delta)$ be a jumping $5' \to 3'$ WK automaton, where $V = \{a_1, \ldots, a_n\}$, and let $w \in V^*$. We define the Parikh vector $o = (o_1, \ldots, o_n)$ of processed (read) symbols from w in a configuration $\gamma = (q, s, w_1, w_2, w_3)$ of M reached from an initial configuration $(q_0, \oplus, \varepsilon, w, \varepsilon)$ of M as $o = \Psi_V(w) - \Psi_V(w_1 w_2 w_3)$, $q \in Q$, $s \in \{\oplus, \ominus\}$, $w_1, w_2, w_3 \in (V \cup \{\#\})^*$. Using the Parikh mapping of $L(M)$, we define $\Delta(o) = \{\sum_{i=1}^{n}(m_i - o_i) : (m_1, \ldots, m_n) \in \Psi_V(L(M)),\ m_i \geq o_i,\ 1 \leq i \leq n\} \cup \{\infty\}$. Finally, we define the *debt* of the configuration γ of M as $\min \Delta(o)$.

Lemma. *Let L be a language, and let $M = (V, Q, q_0, F, \delta)$ be a jumping $5' \to 3'$ WK automaton. If $L(M) = L$, there exists a constant k for M such that M accepts all $w \in L$ using only configurations that have their debt bounded by k.*

Other most notable results of this chapter are summarized in Figure 4.1.

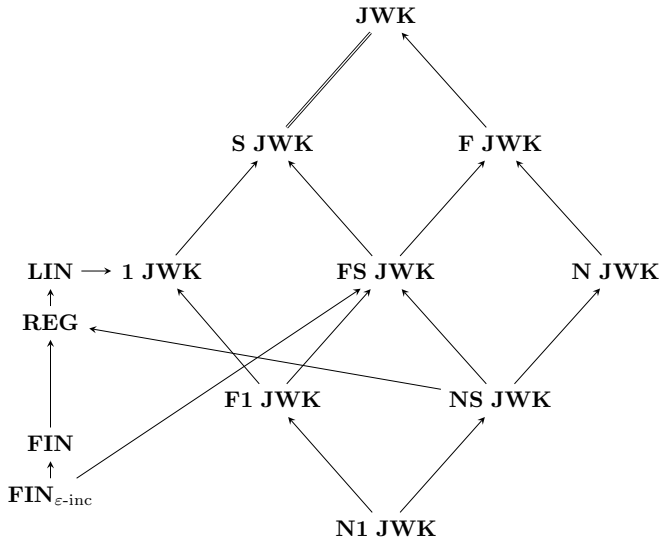

Figure 4.1: A hierarchy of language families closely related to the unrestricted and restricted variants of jumping $5' \to 3'$ WK automata is shown. If there is a double line between families X and Y, then $X = Y$. If there is an arrow from family X to family Y, then $X \subset Y$. Furthermore, if there is no path (following the arrows and double lines) between families X and Y, then X and Y are incomparable.

Chapter 5

General CD Grammar Systems: Normal Forms

This chapter introduces new normal forms of general CD grammar systems fitting for a parallel rewriting process. The content of this chapter is composed of results that were first introduced in a short abstract at the conference AFL 2017 and later published in Journal of Automata, Languages and Combinatorics (see [35]). We concentrate our attention on two-component CD grammar systems working under the $*$ and t modes. Recall that under the former mode the context-free versions of these systems obviously generate only the family of context-free languages. More surprisingly, under the latter mode they are no more powerful than ordinary context-free grammars either. To increase their power, we use general CD grammar systems, whose components are general grammars, that are computationally complete—that is, they characterize the family of recursively enumerable languages. Most importantly, however, we explain how to turn arbitrary general grammars into equivalent two-component general CD grammar systems of very reduced and simplified forms.

To give an insight into this study in a greater detail, take any general grammar G. This chapter demonstrates two types of transformations that turn G into a two-component general CD grammar system with one context-free component and one non-context-free component. For brevity, in this introductory section, Γ_1 and Γ_2 denote the systems resulting from the first type of transformations and the second type of transformations, respectively. Γ_1 has its non-context-free component containing the rules $11 \to 00$ and $0000 \to \varepsilon$, while Γ_2 has its non-context-free component containing the rules $11 \to 00$ and $0000 \to 2222$, where 0, 1, and 2 are new nonterminals. The chapter proves that working under the $*$ and t modes, Γ_1 and Γ_2 are equivalent to G. Thus, more generally speaking, general CD grammar systems of these two forms are computationally complete—that is, they characterize the family of recursively enumerable languages. Apart from the computational completeness, we also explore other useful properties (compactness, single switching of components, close simulation, multi-derivations, and homogeneous rules) which make Γ_1 and Γ_2 simple and easy to apply in theory as well as in practice.

In order to simplify the reasoning for underlying proofs, our first results assume that all input grammars satisfy Kuroda normal form. We start with a two-component general CD grammar system that works in the $*$ mode and has the second component homogeneous.

Theorem. *Let $G = (N, T, P, S)$ be a grammar in Kuroda normal form. Then, there exists a two-component general CD grammar system $\Gamma = (N', T, H, I, S)$ such that H is context-free, $I = \{11 \to 00,\ 0000 \to \varepsilon\}$, and $L_*(\Gamma) = L(G)$.*

Corollary. *The resulting two-component general CD grammar system* Γ *closely simulates the original grammar* G*.*

Next, we consider a two-component general CD grammar system with the same structure but working in the t mode.

Theorem. *Let* $G = (N, T, P, S)$ *be a grammar in Kuroda normal form. Then, there exists a two-component general CD grammar system* $\Gamma = (N', T, H, I, S)$ *such that* H *is context-free,* $I = \{11 \to 00,\ 0000 \to \varepsilon\}$*, and* $L_t(\Gamma) = L(G)$*.*

Corollary. *The resulting two-component general CD grammar system* Γ *changes its components, during every generation of a sentence, no more than once.*

Finally, we change the second component of the two-component general CD grammar system so it is evenly homogeneous. We show that such a system also works correctly in both the $*$ mode and the t mode.

Theorem. *Let* $G = (N, T, P, S)$ *be a grammar in Kuroda normal form. Then, there exists a two-component general CD grammar system* $\Gamma = (N', T, H, I, S)$ *such that* H *is context-free,* $I = \{11 \to 00,\ 0000 \to 2222\}$*, and* $L_*(\Gamma) = L_t(\Gamma) = L(G)$*.*

Corollary. *If the two-component general CD grammar system* Γ *works in the* $*$ *mode, it can closely simulate the original grammar* G*.*

The second part of this chapter considers transformations that turn arbitrary general grammars into equivalent two-component general CD grammar systems. We say that a transformation from general grammars into two-component general CD grammar systems is *direct* if it keeps the original context-free rules intact and splits the non-context-free rules proportionally to the number of symbols on their left-hand sides.

Theorem. *Let* $G = (N, T, P, S)$ *be a general grammar such that* $\mathrm{alph}(\mathrm{lhs}(p)) \cap T = \emptyset$ *for all* $p \in P$*. Then, there exists its direct transformation into a two-component general CD grammar system* $\Gamma = (N', T, H, I, S)$ *such that* H *is context-free,* $I = \{11 \to 00,\ 0000 \to \varepsilon\}$*, and* $L_*(\Gamma) = L_t(\Gamma) = L(G)$*.*

The similar result can be easily achieved for the two-component general CD grammar system where $I = \{11 \to 00,\ 0000 \to 2222\}$. Furthermore, the other properties from the previous section (close simulation and switching of components) still hold in this more general transformation.

Lastly, we introduce a modification of the above transformation that works with all general grammars. However, it is not possible to directly use our previous approach for grammars that have rules with terminals on their left-hand sides. Consequently, the resulting system may not be able to closely simulate the original general grammar. We say that a transformation is *semi-direct* if it separates terminals from the left-hand side of the rules but otherwise behaves as a direct transformation.

Theorem. *Let* $G = (N, T, P, S)$ *be a general grammar. Then, there exists its semi-direct transformation into a two-component general CD grammar system* $\Gamma = (N', T, H, I, S)$ *such that* H *is context-free,* $I = \{11 \to 00,\ 0000 \to \varepsilon\}$*, and* $L_*(\Gamma) = L_t(\Gamma) = L(G)$*.*

The same holds for the system with the evenly homogeneous component.

Chapter 6

Conclusion

In this chapter, we give a short summary and an overview of application and theoretical perspectives that are discussed in the full thesis.

6.1 Application Perspectives

In the core chapters of this thesis, the content is mainly presented in a strictly rigorous way as it appeared in the published papers. In each chapter in the full thesis, we have already mentioned some suggestions for further investigation that are directly linked to the topics in question. Here, we pinpoint some broader application perspectives for the obtained results that are also further described in the full thesis.

First, we discuss the idea of controlled discontinuous reading. We see that, on the one hand, the classical finite automata process the input in a strictly continuous way, and, on the other hand, the original jumping finite automata process the input in a purely discontinous way. We argue that neither of these behaviors alone may be desired for modern computation methods that want to process information discontinously but in a controlled way. If we take a look at the automata introduced in this thesis, we can say that these parallel versions of jumping finite automata explore controlled discontinuous reading behaviors that combine both approaches together.

Second, we explore how to use the presented new proof technique of the debt lemma in a broader context. To demonstrate this in detail we show how to adapt the debt lemma for classical finite automata.

Lastly, we hint how the new special forms of general CD grammar systems can be utilized in practice in a general parallel rewriting process.

6.2 Summary and Theoretical Perspectives

In this last section, we briefly evaluate achieved results and give our final thoughts on the topic. We will not discuss all individual results in detail since each main chapter already has its own concluding remarks in the full thesis, but we will look at the results from a more general perspective.

New Results on Jumping Automata

Considering parallel jumping finite automata, we can see that the newly introduced models match with our goals presented in Chapter 1. All the models fall into the category (PA.2)

of parallelism in multi-head finite automata where the heads cooperate to process the single input. In terms of the general categories for parallelism, these models sort of fall into all of them: In category (P.1), parallelism increases the expressive power of the model. We can see that in Chapter 2 the models clearly extend the original jumping finite automaton and have greater expressive powers. In category (P.2), parallelism is a fundamental part of the behavior of the model. In Chapter 4, we work with Watson-Crick models that fall into (P.2) by definition. Lastly, in category (P.3), parallelism splits the work of the task. Since all the models fall into (PA.2), they also naturally fall into (P.3).

With our work, we have pioneered the study of automaton models that combine the parallel and jumping mechanisms. We believe that this area of research nicely supplements the ongoing thorough investigation of the jumping mechanism. Moreover, our results have already inspired some other new models (see [27]). With our results, we have shown that every additional head increases the power of the model and that these automata can be natural counterparts to various kinds of parallel grammars. In our study of 2-jumps, we have shown that, even if we precisely replicate the behavior of right jumps in left jumps, the right-right and left-left 2-jumps define incomparable language families. We have also studied the possibilities of the combined model of jumping and Watson-Crick finite automata, and our results have introduced some new proof techniques like the debt lemma that can be used even outside the scope of jumping finite automata.

In our research, we have always followed the path that looked the most promising to yield new interesting general results. Nonetheless, there are many possibilities how to combine the parallel and jumping mechanism, and thus also many unexplored areas that we were not able to cover. In the previous chapters of the full thesis, we have already hinted some of the areas with potential for future research. For now, jumping finite automata are still primarily interesting from the theoretical point of view. Indeed, they nicely connect different research areas. But their behavior can be quite wild in more complex cases, their power highly depends on the details of the jumping mechanism, and there are still questions about the complexity of operations with these models. Nonetheless, with enough theoretical knowledge, it can be possible in the future to design and fine tune jumping models so that they properly capture some specific practical problems.

New Results on CD Grammar Systems

Considering the normal forms of grammars and grammar systems, we have introduced several special forms for general CD grammar systems that have interesting parallel properties. The theoretical aspects of these forms are thoroughly described in the full thesis, and we have also hinted some ideas for practical use in the application perspectives. Note that we are not making the generation process of recursively enumerable languages any simpler, we are just rearranging its parts so that it can be run in a parallel way. Therefore, this approach may not be that interesting for the general case, but it can be useful for more specific cases where we can somehow control and guide the generation process.

Compared to jumping models, this area of research is rather isolated and self-contained. We are not aware of other studies of this type in the basic research in the theory of formal languages. Indeed, classical normal forms are primarily focused only on the very restricted forms of rules, minimum number of non-context-free rules, and minimum number of non-terminals. Therefore, we hope that our effort can spark some interest for this largely unexplored research area.

Bibliography

[1] Beier, S. and Holzer, M. Decidability of Right One-Way Jumping Finite Automata. In: *Developments in Language Theory, DLT 2018*. Springer International Publishing, 2018, p. 109–120. LNCS, vol. 11088.

[2] Beier, S. and Holzer, M. Properties of Right One-Way Jumping Finite Automata. In: *Descriptional Complexity of Formal Systems, DCFS 2018*. Springer International Publishing, 2018, p. 11–23. LNCS, vol. 10952.

[3] Beier, S. and Holzer, M. Properties of Right One-Way Jumping Finite Automata. *Theoretical Computer Science*. 2019, vol. 798, p. 78–94.

[4] Beier, S., Holzer, M. and Kutrib, M. Operational State Complexity and Decidability of Jumping Finite Automata. In: *Developments in Language Theory, DLT 2017*. 2017, p. 96–108. LNCS, vol. 10396.

[5] Beier, S., Holzer, M. and Kutrib, M. Operational State Complexity and Decidability of Jumping Finite Automata. *International Journal of Foundations of Computer Science*. 2019, vol. 30, no. 1, p. 5–27.

[6] Chigahara, H., Fazekas, S. Z. and Yamamura, A. One-way Jumping Finite Automata. In: *The 77th National Convention of IPSJ*. 2015.

[7] Chigahara, H., Fazekas, S. Z. and Yamamura, A. One-way Jumping Finite Automata. *International Journal of Foundations of Computer Science*. 2016, vol. 27, no. 03, p. 391–405.

[8] Csuhaj Varjú, E., Dassow, J., Kelemen, J. and Paun, G. *Grammar Systems: A Grammatical Approach to Distribution and Cooperation*. Gordon and Breach Science Publishers, Inc., 1994.

[9] Csuhaj Varjú, E., Masopust, T. and Vaszil, G. Cooperating Distributed Grammar Systems with Permitting Grammars as Components. *Romanian Journal of Information Science and Technology*. 2009, vol. 12, no. 2, p. 175–189.

[10] Csuhaj Varjú, E., Martín Vide, C. and Mitrana, V. Multiset Automata. In: *Multiset Processing*. Springer Berlin Heidelberg, 2001, p. 69–83. LNCS, vol. 2235.

[11] Eğecioğlu, O., Hegedüs, L. and Nagy, B. Stateless Multicounter $5' \to 3'$ Watson-Crick Automata. In: *Fifth IEEE International Conference on Bio-Inspired Computing: Theories and Applications, BIC-TA 2010*. 2010, p. 1599–1606.

[12] Fazekas, S. Z., Hoshi, K. and Yamamura, A. Enhancement of Automata with Jumping Modes. In: *AUTOMATA 2019: Cellular Automata and Discrete Complex Systems*. 2019, p. 62–76. LNCS, vol. 11525.

[13] Fazekas, S. Z. and Yamamura, A. On Regular Languages Accepted by One-Way Jumping Finite Automata. In: *Eighth Workshop on Non-Classical Models of Automata and Applications, NCMA 2016 (Short Papers)*. 2016, p. 7–14.

[14] Fernau, H., Paramasivan, M. and Schmid, M. L. Jumping Finite Automata: Characterizations and Complexity. In: *Implementation and Application of Automata - 20th International Conference, CIAA 2015*. Springer, 2015, p. 89–101. LNCS, vol. 9223.

[15] Fernau, H., Paramasivan, M., Schmid, M. L. and Vorel, V. Characterization and Complexity Results on Jumping Finite Automata. *Theoretical Computer Science*. 2017, vol. 679, p. 31–52.

[16] Geffert, V. Grammars with Context Dependency Restricted to Synchronization. In: *Mathematical Foundations of Computer Science 1986, MFCS 1986*. 1986, p. 370–378. LNCS, vol. 233.

[17] Geffert, V. Context-Free-Like Forms for the Phrase-Structure Grammars. In: *Mathematical Foundations of Computer Science 1988, MFCS 1988*. 1988, p. 309–317. LNCS, vol. 324.

[18] Geffert, V. A Representation of Recursively Enumerable Languages by Two Homomorphisms and a Quotient. *Theoretical Computer Science*. 1988, vol. 62, no. 3, p. 235–249.

[19] Geffert, V. Normal Forms for Phrase-Structure Grammars. *RAIRO-Theor. Inf. Appl.* 1991, vol. 25, no. 5, p. 473–496.

[20] Goldefus, F., Masopust, T. and Meduna, A. Left-Forbidding Cooperating Distributed Grammar Systems. *Theoretical Computer Science*. 2010, vol. 411, 40–42, p. 3661–3667.

[21] Greibach, S. and Hopcroft, J. Scattered Context Grammars. *Journal of Computer and System Sciences*. 1969, vol. 3, no. 3, p. 233–247.

[22] Grune, D. and Jacobs, C. J. *Parsing Techniques: A Practical Guide*. Secondth ed. Springer, 2008.

[23] Hegedüs, L., Nagy, B. and Eğecioğlu, O. Stateless Multicounter $5' \to 3'$ Watson-Crick Automata: The Deterministic Case. *Natural Computing*. 2012, vol. 11, no. 3, p. 361–368.

[24] Holzer, M., Kutrib, M. and Malcher, A. Multi-Head Finite Automata: Characterizations, Concepts and Open Problems. In: *The Complexity of Simple Programs 2008*. 2009, p. 93–107. EPTCS.

[25] Ibarra, O. H. Simple Matrix Languages. *Information and Control*. 1970, vol. 17, p. 359–394.

[26] Immanuel, S. J. and Thomas, D. G. Two-Dimensional Jumping Finite Automata. *Mathematics for Applications*. 2016, vol. 5, no. 2, p. 105–122.

[27] Immanuel, S. J. and Thomas, D. G. Two-Dimensional Double Jumping Finite Automata. *International Journal of Artificial Intelligence and Soft Computing*. 2017, vol. 6, no. 3, p. 250–264.

[28] Inoue, K., Takanami, I., Nakamura, A. and Ae, T. One-Way Simple Multihead Finite Automata. *Theoretical Computer Science*. 1979, vol. 9, no. 3, p. 311–328.

[29] Křivka, Z., Kučera, J. and Meduna, A. Jumping Pure Grammars. *The Computer Journal*. 2018, vol. 62, no. 1, p. 30–41.

[30] Křivka, Z. and Masopust, T. Cooperating Distributed Grammar Systems with Random Context Grammars as Components. *Acta Cybernetica*. 2011, vol. 20, p. 269–283.

[31] Křivka, Z. and Meduna, A. Jumping Grammars. *International Journal of Foundations of Computer Science*. 2015, vol. 26, no. 6, p. 709–731.

[32] Kocman, R. n-Parallel Jumping Finite Automata. In: *Excel@FIT 2015*. 2015.

[33] Kocman, R., Křivka, Z. and Meduna, A. On Double-Jumping Finite Automata. In: *Eighth Workshop on Non-Classical Models of Automata and Applications, NCMA 2016*. Osterreichische Computer Gesellschaft, 2016, p. 195–210. books@ocg.at, vol. 321.

[34] Kocman, R., Křivka, Z. and Meduna, A. On Double-Jumping Finite Automata and Their Closure Properties. *RAIRO-Theor. Inf. Appl.* 2018, vol. 52, 2-3-4, p. 185–199.

[35] Kocman, R., Křivka, Z. and Meduna, A. General CD Grammar Systems and Their Simplification. *Journal of Automata, Languages and Combinatorics*. 2020, vol. 25, no. 1, p. 37–54.

[36] Kocman, R., Křivka, Z., Meduna, A. and Nagy, B. A Jumping $5' \to 3'$ Watson-Crick Finite Automata Model. *Acta Informatica*. (in review).

[37] Kocman, R. and Meduna, A. On Parallel Versions of Jumping Finite Automata. In: *Proceedings of the 2015 Federated Conference on Software Development and Object Technologies, SDOT 2015*. Springer International Publishing, 2016, p. 142–149. Advances in Intelligent Systems and Computing, vol. 511.

[38] Kocman, R., Nagy, B., Křivka, Z. and Meduna, A. A Jumping $5' \to 3'$ Watson-Crick Finite Automata Model. In: *Tenth Workshop on Non-Classical Models of Automata and Applications, NCMA 2018*. Osterreichische Computer Gesellschaft, 2018, p. 117–132. books@ocg.at, vol. 332.

[39] Kudlek, M., Martín-Vide, C. and Păun, G. Toward a Formal Macroset Theory. In: *Multiset Processing*. Springer Berlin Heidelberg, 2001, p. 123–133. LNCS, vol. 2235.

[40] Kudlek, M. and Mitrana, V. Normal Forms of Grammars, Finite Automata, Abstract Families, and Closure Properties of Multiset Languages. In: *Multiset Processing*. Springer Berlin Heidelberg, 2001, p. 135–146. LNCS, vol. 2235.

[41] Kuperberg, D., Pinault, L. and Pous, D. Cyclic Proofs and Jumping Automata. In: *Foundations of Software Technology and Theoretical Computer Science 2019*. 2019.

[42] Kuske, D. and Weigel, P. The Role of the Complementarity Relation in Watson-Crick Automata and Sticker Systems. In: *Developments in Language Theory, DLT 2004*. 2005, p. 272–283. LNCS, vol. 3340.

[43] Loos, R. and Nagy, B. On the Concepts of Parallelism in Biomolecular Computing. *Triangle 6 (Languages: Bioinspired Approaches)*. 2011, p. 109–118.

[44] Madejski, G. Jumping and Pumping Lemmas and Their Applications. In: *Eighth Workshop on Non-Classical Models of Automata and Applications, NCMA 2016 (Short Papers)*. 2016, p. 25–33.

[45] Madejski, G. Regular and Linear Permutation Languages. *RAIRO-Theor. Inf. Appl.* 2018, vol. 52, 2-3-4, p. 219–234.

[46] Madejski, G. and Szepietowski, A. Membership Problem for Two-Dimensional Jumping Finite Automata. In: *Ninth Workshop on Non-Classical Models of Automata and Applications, NCMA 2017 (Short Papers)*. 2017, p. 33–40.

[47] Mahalingam, K., Raghavan, R. and Mishra, U. K. Watson-Crick Jumping Finite Automata. In: *Theory and Applications of Models of Computation, TAMC 2019*. Springer International Publishing, 2019, p. 467–480. LNCS, vol. 11436.

[48] Masopust, T. and Meduna, A. On Pure Multi-Pushdown Automata that Perform Complete Pushdown Pops. *Acta Cybernetica*. 2009, vol. 19, no. 2, p. 537–552.

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

[50] Meduna, A. *Formal Languages and Computation: Models and Their Applications*. Auerbach Publications, 2014.

[51] Meduna, A. and Kolář, D. Homogenous Grammars with a Reduced Number of Non-Context-Free Productions. *Information Processing Letters*. 2002, vol. 81, no. 5, p. 253–257.

[52] Meduna, A. and Masopust, T. Self-Regulating Finite Automata. *Acta Cybernetica*. 2007, vol. 18, no. 1, p. 135–153.

[53] Meduna, A. and Soukup, O. Jumping Scattered Context Grammars. *Fundamenta Informaticae*. 2017, vol. 152, no. 1, p. 51–86.

[54] Meduna, A. and Soukup, O. *Modern Language Models and Computation: Theory with Applications*. Springer, 2017.

[55] Meduna, A. and Švec, M. *Grammars with Context Conditions and Their Applications*. Wiley, 2005.

[56] Meduna, A., Švec, M. and Kopeček, T. Equivalent Language Models that Closely Simulate One Another and Their Illustration in Terms of L Systems. *International Journal of Computer Mathematics*. 2007, vol. 84, no. 11, p. 1555–1566.

[57] Meduna, A. and Zemek, P. Jumping Finite Automata. *International Journal of Foundations of Computer Science*. 2012, vol. 23, no. 7, p. 1555–1578.

[58] Meduna, A. and Zemek, P. *Regulated Grammars and Automata*. Springer, 2014.

[59] Nagy, B., Hegedüs, L. and Eğecioğlu, O. Hierarchy Results on Stateless Multicounter $5' \to 3'$ Watson-Crick Automata. In: *Advances in Computational Intelligence: 11th International Work-Conference on Artificial Neural Networks, IWANN 2011*. Springer, 2011, p. 465–472. LNCS, vol. 6691.

[60] Nagy, B. On $5' \to 3'$ Sensing Watson-Crick Finite Automata. In: *The 13th International Meeting on DNA Computing (DNA13)*. 2007, p. 327–336.

[61] Nagy, B. On $5' \to 3'$ Sensing Watson-Crick Finite Automata. In: *DNA Computing: 13th International Meeting on DNA Computing, DNA13*. Springer, 2008, p. 256–262. LNCS, vol. 4848.

[62] Nagy, B. On a Hierarchy of $5' \to 3'$ Sensing WK Finite Automata Languages. In: *Computability in Europe 2009: Mathematical Theory and Computational Practice, CiE 2009*. 2009, p. 266–275.

[63] Nagy, B. $5' \to 3'$ Sensing Watson-Crick Finite Automata. In: Fung, G., ed. *Sequence and Genome Analysis II – Methods and Applications*. IConcept Press, 2010, p. 39–56.

[64] Nagy, B. A Class of 2-Head Finite Automata for Linear Languages. *Triangle*. 2012, 8 (Languages: Mathematical Approaches), p. 89–99.

[65] Nagy, B. On a Hierarchy of $5' \to 3'$ Sensing Watson-Crick Finite Automata Languages. *Journal of Logic and Computation*. 2013, vol. 23, no. 4, p. 855–872.

[66] Nagy, B. and Kovács, Z. On Simple $5' \to 3'$ Sensing Watson-Crick Finite-State Transducers. In: *Eleventh Workshop on Non-Classical Models of Automata and Applications, NCMA 2019*. 2019, p. 155–170.

[67] Nagy, B. and Otto, F. Two-Head Finite-State Acceptors with Translucent Letters. In: *SOFSEM 2019: Theory and Practice of Computer Science*. Springer International Publishing, 2019, p. 406–418. LNCS, vol. 11376.

[68] Nagy, B. and Otto, F. Linear Automata with Translucent Letters and Linear Context-Free Trace Languages. *RAIRO-Theor. Inf. Appl.* 2020, vol. 54.

[69] Nagy, B. and Parchami, S. On Deterministic Sensing $5' \to 3'$ Watson-Crick Finite Automata: A Full Hierarchy in 2detLIN. *Acta Informatica*. 2020.

[70] Nagy, B., Parchami, S. and Mir-Mohammad-Sadeghi, H. A New Sensing $5' \to 3'$ Watson-Crick Automata Concept. In: *Proceedings 15th International Conference on Automata and Formal Languages, AFL 2017*. Open Publishing Association, 2017, p. 195–204. EPTCS, vol. 252.

[71] Parchami, S. and Nagy, B. Deterministic Sensing $5' \to 3'$ Watson-Crick Automata Without Sensing Parameter. In: *Unconventional Computation and Natural Computation, UCNC 2018*. 2018, p. 173–187. LNCS, vol. 10876.

[72] Păun, G., Rozenberg, G. and Salomaa, A. *DNA Computing: New Computing Paradigms*. Springer-Verlag Berlin Heidelberg, 1998.

[73] Rosebrugh, R. D. and Wood, D. A Characterization Theorem for n-Parallel Right Linear Languages. *Journal of Computer and System Sciences*. 1973, vol. 7, p. 579–582.

[74] Rosebrugh, R. D. and Wood, D. Image Theorems for Simple Matrix Languages and n-Parallel Languages. *Mathematical Systems Theory*. 1974, vol. 8, no. 2.

[75] Rosebrugh, R. D. and Wood, D. Restricted Parallelism and Right Linear Grammars. *Utilitas Mathematica*. 1975, vol. 7, p. 151–186.

[76] Rosenberg, A. L. On Multi-Head Finite Automata. In: *6th Annual Symposium on Switching Circuit Theory and Logical Design, SWCT 1965*. 1965, p. 221–228.

[77] Rozenberg, G. and Salomaa, A. *Handbook of Formal Languages, Vol. 1: Word, Language, Grammar*. Springer-Verlag, 1997.

[78] Rozenberg, G. and Salomaa, A. *Handbook of Formal Languages, Vol. 2: Linear Modeling: Background and Application*. Springer-Verlag, 1997.

[79] Savitch, W. J. How to Make Arbitrary Grammars Look Like Context-Free Grammars. *SIAM Journal on Computing*. 1973, vol. 2, no. 3, p. 174–182.

[80] Siromoney, R. On Equal Matrix Languages. *Information and Control*. 1969, vol. 14, no. 2, p. 135–151.

[81] Siromoney, R. Finite-Turn Checking Automata. *Journal of Computer and System Sciences*. 1971, vol. 5, no. 6, p. 549–559.

[82] Syropoulos, A. Mathematics of Multisets. In: *Multiset Processing*. Springer Berlin Heidelberg, 2001, p. 347–358. LNCS, vol. 2235.

[83] Ďuriš, P. and Hromkovič, J. One-Way Simple Multihead Finite Automata are not Closed Under Concatenation. *Theoretical Computer Science*. 1983, vol. 27, no. 1, p. 121–125.

[84] Vorel, V. Two Results on Discontinuous Input Processing. In: *Descriptional Complexity of Formal Systems: 18th IFIP WG 1.2 International Conference, DCFS 2016*. Springer International Publishing, 2016, p. 205–216. LNCS, vol. 9777.

[85] Vorel, V. Two Results on Discontinuous Input Processing. *Journal of Automata, Languages and Combinatorics*. 2017, vol. 22, 1–3, p. 189–203.

[86] Vorel, V. On Basic Properties of Jumping Finite Automata. *International Journal of Foundations of Computer Science*. 2018, vol. 29, no. 1, p. 1–15.

[87] Wang, Q. and Li, Y. Jumping Restarting Automata. In: *Tenth Workshop on Non-Classical Models of Automata and Applications, NCMA 2018*. Osterreichische Computer Gesellschaft, 2018, p. 181–196. books@ocg.at, vol. 332.

[88] Wood, D. Properties of n-Parallel Finite State Languages. *Utilitas Mathematica*. 1973, vol. 4, p. 103–113.

[89] Wood, D. m-Parallel n-Right Linear Simple Matrix Languages. *Utilitas Mathematica*. 1975, vol. 8, p. 3–28.

[90] Wood, D. n-Linear Simple Matrix Languages and n-Parallel Linear Languages. *Rev. Roum. de Math. Pures et Appl.* 1977, p. 408–412.

[91] Wood, D. *Theory of Computation: A Primer*. Boston: Addison-Wesley, 1987.

Appendix A

Curriculum Vitae

Education

Ph.D. of Computer Science and Engineering 2014 – present
Faculty of Information Technology, Brno University of Technology
Supervisor: prof. RNDr. Meduna Alexander, CSc.

Master of Information Systems 2012 – 2014
Faculty of Information Technology, Brno University of Technology
Master's Thesis: *Dynamic DOM Support in an HTML Rendering Engine*
Supervisor: Ing. Burget Radek, Ph.D.

Bachelor of Information Technology 2009 – 2012
Faculty of Information Technology, Brno University of Technology
Bachelor's Thesis: *Web Application for Ebook Library Management*
Supervisor: Ing. Lengál Ondřej, Ph.D.

Secondary School: Electronic Computer Systems 2005 – 2009
Střední průmyslová škola Jedovnice

Work Experience

Junior Researcher / Technical Staff 2014 – present
Faculty of Information Technology, Brno University of Technology

Web Application Developer 2007 – present
it2b s.r.o., Brno, CZ

Teaching

Compiler Construction — student projects 2018 – 2019

Formal Languages and Compilers — student projects 2014 – 2018

Principles of Programming Languages — student projects 2014 – 2017