Degrees of Free Word-Order and Freely Rewriting Restarting Automata*

F. Mráz[1], **F. Otto**[1], and **M. Plátek**[2]

[1] Fachbereich Mathematik/Informatik, Universität Kassel
34109 Kassel, Germany
{mraz,otto}@theory.informatik.uni-kassel.de

[2] Charles University, Faculty of Mathematics and Physics
Department of Computer Science, Malostranské nám. 25
118 00 PRAHA 1, Czech Republic
platek@ksi.ms.mff.cuni.cz

Abstract. In natural languages with a high degree of word-order freedom syntactic phenomena like dependencies (subordinations) or valencies do not depend on the word-order (or on the individual positions of the individual words). This means that some permutations of sentences of these languages are in some (important) sense syntactically equivalent. Here we study this phenomenon in a formal way. Various types of j-monotonicity for restarting automata can serve as parameters for the degree of word-order freedom and for the complexity of word-order in sentences (languages). Here we combine two types of parameters on computations of restarting automata:

1. the degree of j-monotonicity, and
2. the number of rewrites per cycle.

We study these notions formally in order to obtain an adequate tool for modelling and comparing formal descriptions of (natural) languages with different degrees of word-order freedom and word-order complexity.

1 Introduction

The original motivation for introducing the restarting automaton was the desire to model the so-called *analysis by reduction* of natural languages. Many aspects of the work on restarting automata are motivated by the basic tasks of computational linguistics. (e.g., devising multilevel language descriptions) as well as by applied tasks (e.g., constructing grammar checkers for free word-order languages). More about the motivation and about the corresponding literature can be found in [13, 15].

From a theoretical point of view the restarting automaton can be seen as a tool that yields a very flexible generalization of analytical grammars, that is, in a

* The work of the first two authors was supported by a grant from the Deutsche Forschungsgemeinschaft. The third author was partially supported by the Grant Agency of the Czech Republic under Grant-No. 201/04/2102 and by the program 'Information Society' under project 1ET100300517.

very flexible way it introduces a basic syntactic system (an approximation to the formalization of the analysis by reduction), which contains the full information about the input vocabulary (set of wordforms), the categorial vocabulary, the set of reductions (rewritings), the recognized language, the language of sentential forms, and the categorial language. On the other hand, the restarting automaton can be considered as a generalization and a refinement of the pushdown automaton (see [14]).

Various restricted versions of the restarting automaton and various constraints for it are considered in the literature. In particular, a *monotonicity* constraint has been introduced for restarting automata which is based on the idea that from one rewrite operation to the next within a computation, the actual place where the rewriting is performed must not increase its distance from the *right* end of the tape. The monotone restarting automata essentially model bottom-up context-free analyzers. Accordingly, it was shown that the monotone restarting automata (with auxiliary symbols) characterize exactly the class CFL of context-free languages. and various restricted versions of deterministic monotone restarting automata (with or without auxiliary symbols) characterize the class DCFL of deterministic context-free languages [9].

Also a generalization of the constraint of monotonicity has been considered, which models the generalization from bottom-up one-pass parsers to bottom-up multi-pass parsers. For an integer $j \geq 1$, a computation is called *j-monotone* if the corresponding sequence of rewrite steps can be partitioned into at most j subsequences such that each of these subsequences is monotone. It was shown that by increasing the value of the parameter j, the expressive power of restarting automata without auxiliary symbols is increased [16].

Here we introduce and use a new type of restarting automaton, the *freely rewriting restarting automaton*, FRR-automaton for short, which may perform an unlimited number of rewrite operations per cycle, in order to classify the word-order of natural languages. In fact, we propose an infinite hierarchy of classes of basic syntactic systems and of classes of languages of sentential forms (of restarting automata) from the points of view of word-order complexity and word-order freedom. Our main goal is to show that the word-order freedom (together with some type of valencies (dependencies)) can cause complex syntactic phenomena. Similar considerations were made before using dependency grammars [8], however our approach is more general, and it can be applied to various types of grammars, including Chomsky, categorial, pure, Marcus and dependency grammars, using simulations by restarting automata. The word-order constraints play an important role in modern computational linguistics (see, e.g., [5]). We only consider restarting automata, which are not stronger (with respect to their weak generative capacity) than linear-bounded automata. In order to obtain the intended results certain combinations of constraints for FRR-automata are studied.

The complexity of word-order of natural languages can be illustrated by several constructions found in some languages. The first two samples show non-context-free constructions in languages which are considered to be languages

with fixed word-order. The next three samples are taken from languages with a considerable degree of word-order freedom.

In [3] Bresnan et al. give the following example from Dutch:

(dat) Jan Piet Marie de kinderen zag helpen laten zwemmen.
[*(that)-Jan-Piet-Marie-the-children-saw-help-make-swim.*]
[*(that) Jan saw Piet help Marie make the children swim.*]

which shows a duplication-like structure $w\overline{w}$, where $\overline{w}$ is the word obtained from w by replacing each symbol by its barred copy. Using analysis by reduction we would like to get the following sequence of reductions:

(dat) Jan Piet Marie **de kinderen** zag helpen **laten zwemmen**.
(dat) Jan Piet **Marie** zag **helpen** $\mathbf{VG}_{inf}$.
(dat) Jan Piet zag VG_{dat}.

where the rewritten parts are in bold font. In the first step, the noun phrase 'de kinderen' (*the children*) and the infinitival verb complement 'laten zwemmen' are replaced by the category VG_{inf}, which means that an infinitive construction was deleted. Note that the rewritten part is not contiguous. In the second step, the words 'Marie', 'helpen' and the category VG_{inf} are replaced by a category VG_{dat} which represents a subordinate clause.

Similar constructions, where an adequate analysis by reduction requires rewriting words which are in distant parts of a sentence can be found in many other languages. Shieber found the following construction in Zürich dialect of German [18]:

Jan säit das mer d'chind em Hans es huus haend wele laa hälfe aastrüche.
[*Jan-said-that-we-the-children-Hans-the-house-wanted-to-let-help-paint.*]
[*Jan said that we wanted to let the children help Hans paint the house.*]

It has the structure $xwa^m b^n y c^m d^n z$, where a, b stand for dative and accusative noun phrases, respectively, and c, d for the corresponding dative and accusative verb phrases, respectively.

Analysis by reduction for the above two sample sentences (more precisely their generalized forms) can be modelled by 2-monotone FRR-automata with two rewrites per cycle which we call 2-*constrained*. The degree of j-constrainability will serve as a synonym for word-order complexity. In this way we enrich the taxonomy of various word-order constraints given in [8].

In German the following are correct (parts of) sentences [17]:

- ... daß Peter dem Kunden den Kühlschrank zu reparieren zu helfen versucht.
  [*... that-Peter-the-client-the-refrigerator-to-repair-to-help-tries.*]
  [*... that Peter tries to help the client to repair the refrigerator.*]
- ... daß Peter versucht, dem Kunden zu helfen, den Kühlschrank zu reparieren.

– ... daß Peter versucht, dem Kunden den Kühlschrank zu reparieren zu helfen.

Even more correct sentences can be obtained by permuting five elements (delimited by < and >) in the following example taken from [7]:

... daß <eine hiesige Firma> <meinem Onkel> <die Möbel> <vor drei Tagen> <ohne Voranmeldung> zugestellt hat.
[*... that-<a-local-company>-<my-uncle>-<the-furniture>-<three-days-ago>-<without-advance-notice>-delivered-has.*]
[*... that a local company delivered the furniture to my uncle three days ago without advance notice.*]

For a more detailed discussion on word-order of German see [17].

In some other languages like most Slavonic languages, the syntactic relations between words are specified by means other than the position in a sentence – by a rich inflexion. Often all the permutations of words in a clause are possible[3]. All the following sentences are correct in Czech:

– Všichni prošli kursem.
[*All-passed-the course.*]
– Kursem všichni prošli.
– Kursem prošli všichni.
– Prošli všichni kursem.
– Prošli kursem všichni.
– Všichni kursem prošli.

In the last three examples some words in each sentence can be (almost) freely permuted. This could resemble the transformational grammar approach [4], where the transformations are used to increase the descriptive power of context-free grammars. We will deal with permutations only and say that all sentences which differ in the word-order only are Parikh equivalent (that is, they contain the same number of occurrences of each word). Based on the Parikh equivalence we introduce a reduction equivalence of sentences and study the degree of word-order freedom of languages.

The paper is structured as follows. In the next section we present definitions of FRR-automata and their j-constrained versions. The language $L_{SF}(M)$ accepted by an FRR-automaton M is called a language of sentential forms. If we restrict the set of possible input symbols to a fixed subset Σ of the working alphabet, we say that the automaton M accepts the input language $L_I = L_{SF} \cap \Sigma^*$. In Section 3 we show the power of j-constrained FRR-automata: the languages of sentential forms accepted by j-constrained FRR-automata create an infinite hierarchy. Further, for each degree j of constrainability, the class of languages of sentential forms accepted by j-constrained FRR-automata is a proper subclass of the class of input languages accepted by j-constrained FRR-automata. In other

[3] While the permutations mostly share the same basic meaning represented by dependencies, they nevertheless differ in the topic-focus articulation.

words: the use of auxiliary (non-input) symbols can hide the real complexity of the word-order. Then we formalize the degree of word-order freedom of a language (or an FRR-automaton) through the notion of *j-scalability*. Roughly speaking, an FRR-automaton is j-scalable if increasing constrainability of its computations from i to $i+1$, for all $1 \leq i < j$, extends the set of accepted words only by words obtained by permutations of symbols in words accepted already by i-constrained computations and simultaneously the structure of reductions defined by cycles of M remains the same (up to the Parikh equivalence). Finally, in Section 4 we shortly discuss the achieved results and state some problems for further research on this subject.

2 Definitions

Throughout the paper we will use λ to denote the empty word. Further, $|w|$ will denote the *length* of the word w, and if a is an element of the underlying alphabet, then $|w|_a$ denotes the *a-length* of w, that is, the number of occurrences of the letter a in w. Further, $\mathbb{N}_+$ will denote the set of all positive integers.

We start by describing the model of the restarting automaton we are going to use in this paper.

A *freely rewriting restarting automaton*, FRR-automaton for short, is a (nondeterministic) machine that is described by a 7-tuple $M = (Q, \Gamma, ¢, \$, q_0, k, \delta)$, where Q is a finite set of states, Γ is a finite tape alphabet, $¢, \$ \notin \Gamma$ are symbols that are used as markers for the left and right border of the work space, respectively, $q_0 \in Q$ is the initial state, $k \geq 1$ is the size of the *read/write window*, and δ is the *transition relation* that associates to each pair (q, u) consisting of a state q and a possible content u of the read/write window a finite set of possible transition steps. There are four types of transition steps:

1. A *move-right step* (MVR) causes M to shift the read/write window one position to the right and to change the state. However, the read/write window cannot move across the right sentinel \$.
2. A *rewrite step* causes M to replace a non-empty prefix u of the content of the read/write window by a string v, thereby possibly changing the length of the tape, and to change the state. Further, the read/write window is placed immediately to the right of the string v. However, occurrences of the delimiters ¢ and \$ can neither be deleted nor newly created by a rewrite step.
3. A *restart step* causes M to place its read/write window over the left end of the tape, so that the first symbol it sees is the left sentinel ¢, and to reenter the initial state q_0.
4. An *accept step* causes M to halt and accept.

If $\delta(q, u) = \emptyset$ for some pair (q, u), then M necessarily halts, and we say that M *rejects* in this situation. In addition, it is required that there exists a *weight function* $\omega : \Gamma \to \mathbb{N}_+$ such that, for each rewrite operation $u \to v$, $\omega(v) < \omega(u)$ holds. Here ω is extended to a morphism $\omega : \Gamma^* \to \mathbb{N}$ by taking $\omega(\lambda) := 0$ and

$\omega(wa) := \omega(w) + \omega(a)$ for all $w \in \Gamma^*$ and all $a \in \Gamma$. Thus, the FRR-automaton is a variant of the *shrinking restarting automaton* considered in [12].

A *configuration* of M is a string $\alpha q \beta$ where $q \in Q$, and either $\alpha = \lambda$ and $\beta \in \{¢\} \cdot \Gamma^* \cdot \{\$\}$ or $\alpha \in \{¢\} \cdot \Gamma^*$ and $\beta \in \Gamma^* \cdot \{\$\}$; here q represents the current state, $\alpha\beta$ is the current content of the tape, and it is understood that the window contains the first k symbols of β or all of β when $|\beta| \leq k$. A *restarting configuration* is of the form $q_0 ¢ w \$$.

We observe that any computation of M consists of certain phases. A phase, called a *cycle*, starts in a restarting configuration, the head moves along the tape performing move-right and rewrite operations until a restart operation is performed and thus a new restarting configuration is reached. If no further restart operation is performed, the computation necessarily finishes in a halting configuration – such a phase is called a *tail*. It is required that in each cycle M performs at least one rewrite step – thus each cycle reduces the weight of the actual tape content with respect to the weight function ω mentioned above. Further, it is required that M does not perform any rewrite steps in a tail. We use the notation $u \vdash_M^c v$ to denote a cycle of M that begins with the restarting configuration $q_0 ¢ u \$$ and ends with the restarting configuration $q_0 ¢ v \$$; the relation $\vdash_M^{c^*}$ is the reflexive and transitive closure of $\vdash_M^c$. The pair $\mathsf{RS}(M) := (\Gamma^*, \vdash_M^c)$ is called the *reduction system* induced by M.

A *sentential form* $w \in \Gamma^*$ *is accepted by* M, if there is an accepting computation which starts from the restarting configuration $q_0 ¢ w \$$. By $L_{SF}(M)$ we denote the language consisting of all sentential forms accepted by M; we say that $L_{SF}(M)$ is the *language of sentential forms* recognized by M.

From the above description it is easily concluded that, starting from a configuration of the form $q_0 ¢ w \$$, M will execute at most $c \cdot |w|$ many cycles for some constant $c \in \mathbb{N}_+$, which implies that $L_{SF}(M)$ is accepted by a nondeterministic Turing machine simultaneously in quadratic time and in linear space, that is, $L_{SF}(M) \in \mathsf{NP} \cap \mathsf{CSL}$.

If a proper subalphabet Σ of Γ is fixed as an alphabet of *terminal symbols* (or *input symbols*), then the language $L_I(M) := L_{SF}(M) \cap \Sigma^*$ of all input sentences accepted by M is called the *input language recognized* by M. In this case the four-tuple $\mathsf{BS}_I(M) := (\Sigma, \Gamma, L_{SF}(M), \mathsf{RS}(M))$ is called the *basic syntactic system* of M with input alphabet Σ. Observe that $\mathsf{BS}_I(M)$ contains the complete information about $L_I(M), L_{SF}(M)$, and $\mathsf{RS}(M)$, and that $L_{SF}(M)$ plays the main role among the two languages. In an obvious way one can introduce basic syntactic systems also for Chomsky grammars and various types of categorial and dependency grammars.

We emphasize the following nice properties of restarting automata, which are often used implicitly in proofs. Observe that the latter property does in general not hold for input languages.

Fact 1 (Error Preserving Property). *Let M be an* FRR*-automaton, and let $u, v \in \Gamma^*$. If $u \vdash_M^{c^*} v$ and $u \notin L_{SF}(M)$, then $v \notin L_{SF}(M)$, either.*

Fact 2 (Correctness Preserving Property). *Let M be a an* FRR*-automaton, and let $u, v \in \Gamma^*$. If $u \vdash_M^{c^*} v$ is a part of an accepting computation of M, then $v \in L_{SF}(M)$.*

Finally we come to the notion of *monotonicity*. Let $C := \alpha q \beta$ be a *rewrite configuration* of a FRR-automaton M, that is, a configuration in which a rewrite instruction is to be applied. Then $|\beta|$ is called the *right distance* of C, which is denoted by $D_r(C)$. A *sequence of rewrite configurations* $S = (C_1, C_2, \ldots, C_n)$ is called *monotone* if $D_r(C_1) \geq D_r(C_2) \geq \ldots \geq D_r(C_n)$.

Let j be a positive integer. We say that a sequence of rewrite configurations $S = (C_1, C_2, \ldots, C_n)$ is *j-monotone* if there is a partition of S into j subsequences

$$S_1 = (C_{1,1}, C_{1,2}, \cdots, C_{1,p_1}), \ldots, S_j = (C_{j,1}, C_{j,2}, \cdots, C_{j,p_j}),$$

such that each S_i, $1 \leq i \leq j$, is monotone. Obviously a sequence of rewrite configurations $(C_1, C_2, \cdots, C_n)$ is *not* j-monotone if and only if there exist indices $1 \leq i_1 < i_2 < \ldots < i_{j+1} \leq n$ such that $D_r(C_{i_1}) < D_r(C_{i_2}) < \ldots < D_r(C_{i_{j+1}})$.

A *computation* of an FRR-automaton M is called *j-monotone* if the corresponding sequence of rewrite configurations is j-monotone. A *computation* is *j-rewriting* if none of its cycles contains more than j rewrite steps. Finally, a *computation* is *j-constrained* if it is both j-rewriting and j-monotone.

Notation. For any class A of automata and any index $Y \in \{I, SF\}$, $\mathcal{L}_Y(\mathsf{A})$ will denote the class of Y-languages recognizable by automata from A, and $\mathcal{BS}_I(\mathsf{A})$ will denote the class of basic syntactic systems determined by A. By (D)CFL we denote the class of (deterministic) context-free languages, and by $\subset$ we denote the proper subset relation. Sometimes we will use regular expressions instead of the corresponding regular languages.

3 Constraints of word-order

In this section we introduce some constraints which will be used for characterizations of word-order complexity and word-order freedom. For us the degree of word-order freedom means a certain degree of robustness against permutations of sentential forms and their reductions, while the degree of constrainability serves as a synonym for the degree of word-order complexity. In the following we will restrict our attention mainly to languages of sentential forms. First we introduce an infinite hierarchy of classes of languages of sentential forms based on the notion of constrainability.

Definition 3. *For an* FRR*-automaton M over Γ, and an integer $i \in \mathbb{N}_+$,*

$$L_{SF}(M, i) := \{ w \in L_{SF}(M) \mid M \text{ accepts } w \text{ by an } i\text{-constrained computation} \}.$$

In addition, if an input alphabet $\Sigma \subset \Gamma$ is fixed for M, then

$$L_I(M, i) := \{ w \in L_I(M) \mid M \text{ accepts } w \text{ by an } i\text{-constrained computation} \}.$$

For each FRR-automaton M, each input alphabet Σ, each $Y \in \{I, SF\}$, and each $i \geq 1$, $L_Y(M,i) \subseteq L_Y(M,i+1)$, as each i-constrained computation is also $(i+1)$-constrained.

Definition 4. *Let $i \in \mathbb{N}_+$, and let $\mathcal{A}$ be a type of restarting automaton. Then*

$$\mathcal{L}_{SF}(i,\mathcal{A}) := \{\, L_{SF}(M,i) \mid M \text{ is an } \mathcal{A}\text{-automaton} \,\}.$$

Further, by

$$\mathcal{L}_{I}(i,\mathcal{A}) := \{\, L_{I}(M,i) \mid M \text{ is an } \mathcal{A}\text{-automaton} \,\}$$

we denote the corresponding class of languages obtained when considering $\mathcal{A}$-automata with designated input alphabets.

The next theorem follows from results presented in [9].

Theorem 1. $\mathsf{DCFL} \subset \mathcal{L}_{SF}(1,\mathsf{FRR}) \subset \mathsf{CFL} = \mathcal{L}_{I}(1,\mathsf{FRR})$.

Proof. In [11] the *shrinking* restarting automaton is introduced (see also [12]). It differs from the standard restarting automaton in that rewrite steps of the form $u \to v$ are not necessarily length-reducing, but that they are only required to be weight-reducing with respect to some weight function. Each monotone shrinking restarting automaton without auxiliary symbols (that is, a monotone shrinking RRW-automaton in the notation of [14]) can be seen as an FRR-automaton that executes only 1-constrained computations. Thus, $\mathcal{L}_{SF}(1,\mathsf{FRR}) = \mathcal{L}(\text{mon-sRRW})$. Hence, we obtain the proper inclusions $\mathsf{DCFL} \subset \mathcal{L}_{SF}(1,\mathsf{FRR}) \subset \mathsf{CFL}$ from the results of [9]. While in [9] only restarting automata are considered for which each rewrite step is length-reducing, it is easily seen that the context-free example language given there which is not accepted by any RRW-automaton is not accepted by any shrinking RRW-automaton, either.

The language class CFL coincides with the class of languages that are accepted by monotone RRWW-automata. In fact, it coincides with the class of languages accepted by monotone shrinking RRWW-automata [11]. However, it is easily seen that monotone shrinking RRWW-automata correspond to FRR-automata with designated input alphabets executing only 1-constrained computations. This yields the equality $\mathsf{CFL} = \mathcal{L}_{I}(1,\mathsf{FRR})$. □

Next we introduce a restricted type of FRR-automaton called MRR-*automaton*. The transition relation of an MRR-automaton M is given through a finite sequence of so-called *meta-instructions* of the form

$$(E_1, u_1 \to v_1, E_2, u_2 \to v_2, E_3, \ldots, E_i, u_i \to v_i, E_{i+1}),$$

where $E_1, \ldots, E_{i+1}$ are regular expressions, and for each $n = 1, \ldots, i$, $u_n, v_n \in \Gamma^*$ are strings satisfying $\omega(v_n) < \omega(u_n)$, where $\omega : \Gamma \to \mathbb{N}_+$ is a weight function associated with M. The rules $u_1 \to v_1$, $u_2 \to v_2$, …, $u_i \to v_i$ embody rewrite steps of M. On trying to execute this meta-instruction, M will get stuck (and so reject) starting from the configuration $q_0 ¢ w \$$, if w does not admit a factorization of the form $w = w_1 u_1 w_2 u_2 \ldots w_i u_i w_{i+1}$ such that $¢ w_1 \in L(E_1)$, $w_2 \in L(E_2)$, …,

$w_{i+1}\$ \in L(E_{i+1})$, where $L(E_n)$ denotes the language described by the regular expression E_n. On the other hand, if w does have a factorization of this form, then one such factorization is chosen nondeterministically, and $q_0 ¢ w \$$ is transformed into $q_0 ¢ w_1 v_1 w_2 v_2 w_3 \ldots w_i v_i w_{i+1} \$$. In order to describe the tails of accepting computations of M (during which M cannot apply any rewrite operations at all), we use meta-instructions of the form $(¢ \cdot E \cdot \$, \mathsf{Accept})$, which accepts the sentences from the regular language $L(E)$.

For each MRR-automaton M, there exists a number j_M such that no cycle of any computation of M contains more than j_M applications of rewrite steps, that is, each computation of M is j_M-rewriting. Thus, MRR-automata have *limited rewriting* only. In this sense MRR-automata are weaker than FRR-automata.

Example 1. We consider the language $L_\infty := \{ (a^n b^n)^m \mid n, m \geq 0 \}$. It is easily seen that the following deterministic FRR-automaton M_∞ with initial state q_0 and tape alphabet $\Gamma := \{a, b\}$ accepts this language:

$$
\begin{array}{ll}
\delta(q_0, ¢\$) = \mathsf{Accept}, & \delta(q_1, aaa) = (q_0, \mathsf{MVR}),\\
\delta(q_0, ¢aa) = (q_0, \mathsf{MVR}), & \delta(q_1, aab) = (q_0, \mathsf{MVR}),\\
\delta(q_0, aaa) = (q_0, \mathsf{MVR}), & \delta(q_1, bb\$) = \mathsf{Restart},\\
\delta(q_0, aab) = (q_0, \mathsf{MVR}), & \delta(q_1, b\$) = \mathsf{Restart},\\
\delta(q_0, abb) = (q_1, b), & \delta(q_1, \$) = \mathsf{Restart},\\
\delta(q_1, bbb) = (q_1, \mathsf{MVR}), & \delta(q_0, ¢ab) = (q_2, ¢),\\
\delta(q_1, bba) = (q_1, \mathsf{MVR}), & \delta(q_2, ab) = (q_2, \lambda),\\
\delta(q_1, baa) = (q_0, \mathsf{MVR}), & \delta(q_2, \$) = \mathsf{Accept}.
\end{array}
$$

Given the string $w = (a^3 b^3)^4$ as input, M will execute the following computation:

$$
\begin{aligned}
q_0 ¢ a^3 b^3 (a^3 b^3)^3 \$ &\vdash_M^* ¢ a^2 q_0 abbb (a^3 b^3)^3 \$\\
&\vdash_M ¢ a^2 b q_1 b (a^3 b^3)^3 \$\\
&\vdash_M^* ¢ a^2 b^2 a^2 q_0 abbb (a^3 b^3)^2 \$\\
&\vdash_M ¢ a^2 b^2 a^2 b q_1 b (a^3 b^3)^2 \$\\
&\vdash_M^* ¢ (a^2 b^2)^2 a^2 q_0 abbb a^3 b^3 \$\\
&\vdash_M ¢ (a^2 b^2)^2 a^2 b q_1 b a^3 b^3 \$\\
&\vdash_M^* ¢ (a^2 b^2)^3 a^2 q_0 abbb \$\\
&\vdash_M ¢ (a^2 b^2)^3 a^2 b q_1 b \$\\
&\vdash_M q_0 ¢ aabb (a^2 b^2)^3 \$\\
&\vdash_M^* ¢ a q_0 abb (a^2 b^2)^3 \$\\
&\vdash_M ¢ ab q_1 aabb (a^2 b^2)^2 \$\\
&\vdash_M ¢ aba q_0 abb (a^2 b^2)^2 \$\\
&\vdash_M ¢ ababq_1 aabba^2 b^2 \$\\
&\vdash_M ¢ ababaq_0 abba^2 b^2 \$\\
&\vdash_M ¢ abababq_1 aabb \$\\
&\vdash_M ¢ ababababaq_0 abb \$\\
&\vdash_M ¢ abababababq_1 \$\\
&\vdash_M q_0 ¢ abababab \$\\
&\vdash_M ¢ q_2 ababab \$\\
&\vdash_M^* ¢ q_2 \$ \vdash_M \mathsf{Accept}.
\end{aligned}
$$

On the other hand, if M' is an MRR-automaton on Γ, then M' has no accepting computation for an input of the form $w := (a^n b^n)^{j_{M'}+1}$, where n is sufficiently large. Indeed, as M' can execute at most $j_{M'}$ rewrite steps per cycle, the first cycle of M' in an accepting computation on input w will transform the string w into a string not belonging to the language L_∞, thus violating the Correctness Preserving Property.

Thus, we have the following consequence.

Corollary 1. $\mathcal{L}_{SF}(\mathsf{MRR}) \subset \mathcal{L}_{SF}(\mathsf{FRR})$.

For the FRR-automaton M of Example 1 we see that

$$L_{SF}(M,i) = \{\, (a^n b^n)^m \mid n \geq 0,\, m \leq i \,\},$$

which shows that the languages $L_{SF}(M,i)$ $(i \geq 1)$ form an infinite strictly ascending sequence approximating the language $L_{SF}(M) = L_\infty$. Again the language $\{\, (a^n b^n)^m \mid n \geq 0,\, m \leq i \,\}$ cannot be accepted by any FRR-automaton performing only $(i-1)$-constrained computations. Hence, we have the following infinite hierarchy.

Corollary 2.

(a) *For all* $i \geq 1$, $\mathcal{L}_{SF}(i, \mathsf{FRR}) \subset \mathcal{L}_{SF}(i+1, \mathsf{FRR})$.

(b) $\bigcup_{i \geq 1} \mathcal{L}_{SF}(i, \mathsf{FRR}) \subset \mathcal{L}_{SF}(\mathsf{FRR})$.

Example 2. Let M_1 be the MRR-automaton with tape alphabet $\{a, b\}$ that is given through the following set of meta-instructions:

(1) $(\mathfrak{c} \cdot a^+, abb \to b, b^* \cdot \$)$,
(2) $(\mathfrak{c} \cdot a^+, abb \to b, b^* \cdot a^+, abb \to b, b^* \cdot \$)$,
(3) $(\mathfrak{c} \cdot (ab + abab) \cdot \$,\ \mathsf{Accept})$.

It is easy to see that $L_{SF}(M_1, 1) = \{\, a^n b^n \mid n > 0 \,\} \cup \{abab\}$, as in 1-constrained computations M_1 can only use meta-instructions (1) and (3), while $L_{SF}(M_1, 2) = L_{SF}(M_1, 1) \cup \{\, a^n b^n a^n b^n \mid n > 0 \,\}$. Hence, $L_{SF}(M_1) = L_{SF}(M_1, 2) \supset L_{SF}(M_1, 1)$.

Remark 1. Observe that $L_{SF}(M_1) = L_{SF}(M_1, 2) = \{\, a^n b^n, a^n b^n a^n b^n \mid n > 0 \,\}$ is not context-free, and so $L_{SF}(M_1) \notin \mathcal{L}_{SF}(1, \mathsf{FRR})$ by Theorem 1.

Example 3. Let M_2 be the MRR-automaton with the same tape alphabet as M_1 that is given through the meta-instructions of M_1 and the following two instructions:

$$(4)\ (\mathfrak{c} \cdot a^+, abb \to b, b^* \cdot ab \cdot \$),\ (5)\ (\mathfrak{c} \cdot ab \cdot a^+, abb \to b, b^* \cdot \$).$$

In 1-constrained computations M_2 can use the meta-instructions (1) and (3) to (5). It follows easily that $L_{SF}(M_2, 1) = \{\, a^n b^n, aba^n b^n, a^n b^n ab \mid n > 0 \,\}$, while $L_{SF}(M_2, 2) = \{\, a^n b^n a^m b^m \mid n > 0,\, m \geq 0 \,\} = L_{SF}(M_2)$.

In contrast to $L_{SF}(M_1)$, the language $L_{SF}(M_2)$ is context-free, and $L_{SF}(M_2)$ can even be accepted by an MRR-automaton M' that executes only 1-constrained computations, that is, $L_{SF}(M_2) \in \mathcal{L}_{SF}(1, \mathsf{FRR})$. M' is given through the following meta-instructions:

(1) $(\mathfrak{c} \cdot a^+, abb \to b, b^* \cdot \$)$, (3) $(\mathfrak{c} \cdot a^+, abb \to b, b^* \cdot a^+ \cdot b^+ \cdot \$)$,
(2) $(\mathfrak{c} \cdot ab \to \lambda, a^+ \cdot b^+ \cdot \$)$, (4) $(\mathfrak{c} \cdot ab \cdot \$, \mathsf{Accept})$.

Below we will use the following families of sample languages, where $j \in \mathbb{N}_+$, $\Gamma_j := \{a, b, c_0, c_1, \ldots, c_{j-1}\}$, and $\Delta_j := \{a_1, a_2, \ldots, a_j\}$:

$$LC_j := \{\, c_0 w c_1 w \ldots c_i w \ldots c_{j-2} w c_{j-1} w \mid w \in \{a,b\}^* \,\} \subset \Gamma_j^*,$$
$$LE_j := \{\, w \in \{a_1, ..., a_j\}^* \mid |w|_{a_1} = |w|_{a_2} = ... = |w|_{a_j} \,\} \subset \Delta_j^*.$$

Let us remark that the languages LC_j $(j \geq 1)$ represent those languages in which valencies are modelled through fixed reduction positions (with an increasing number of such positions), and the languages LE_j $(j \geq 1)$ represent those languages in which valencies are modelled by j-tuples of symbols in varying positions, independent of the word-order.

Example 4. Let $j \geq 1$, and let MC_j be the MRR-automaton with tape alphabet Γ_j that is given through the following sequence of meta-instructions, where $\Sigma_0 := \{a, b\}$:

(1) $(\mathfrak{c}c_0, a \to \lambda, \Sigma_0^* \cdot c_1, a \to \lambda, \Sigma_0^* \cdot c_2, \ldots, \Sigma_0^* \cdot c_{j-1}, a \to \lambda, \Sigma_0^* \cdot \$)$;
(2) $(\mathfrak{c}c_0, b \to \lambda, \Sigma_0^* \cdot c_1, b \to \lambda, \Sigma_0^* \cdot c_2, \ldots, \Sigma_0^* \cdot c_{j-1}, b \to \lambda, \Sigma_0^* \cdot \$)$;
(3) $(\mathfrak{c}c_0 c_1 \ldots c_{j-1}\$, \mathsf{Accept})$.

It follows immediately that $L_{SF}(MC_j) = LC_j$ holds, and that MC_j is j-constrained. On the other hand, it is easily seen that $LC_{j+1} \notin \mathcal{L}_{SF}(j, \mathsf{FRR})$, as for a sentence $x \in LC_{j+1}$, j rewrite operations per cycle do in general not suffice to transform x into another string that still belongs to LC_{j+1}. Hence, LC_{j+1} does not coincide with the language of sentential forms for any j-constrained FRR-automaton.

However, for each $j \geq 1$, there exists an MRR-automaton $\overline{M}_j$ with input alphabet Γ_j and tape alphabet $\Omega_j := \Gamma_j \cup D$, where $D := \{d_a, d_b\}$, such that $L_I(\overline{M}_j, 2) = LC_j$. The automaton $\overline{M}_j$ is given through the following meta-instructions, where $u \in \Sigma_0$:

(1) $(\mathfrak{c}c_0, u \to \lambda, \Sigma_0^* \cdot c_1 \cdot D^*, u \to d_u, \Sigma_0^* \cdot c_2 \cdot \ldots \cdot c_{j-1} \cdot \Sigma_0^* \cdot \$)$,
(2) $(\mathfrak{c}c_0 c_1, d_u \to \lambda, D^* \cdot c_2 \cdot D^*, u \to d_u, \Sigma_0^* \cdot c_3 \cdot \ldots \cdot c_{j-1} \cdot \Sigma_0^* \cdot \$)$,
...
(j − 2) $(\mathfrak{c}c_0 c_1 \ldots c_{j-3}, d_u \to \lambda, D^* \cdot c_{j-2} \cdot D^*, u \to d_u, \Sigma_0^* \cdot c_{j-1} \cdot \Sigma_0^* \cdot \$)$,
(j − 1) $(\mathfrak{c}c_0 c_1 \ldots c_{j-2}, d_u \to \lambda, D^* \cdot c_{j-1}, u \to \lambda, \Sigma_0^* \cdot \$)$,
(j) $(\mathfrak{c}c_0 c_1 \ldots c_{j-1}\$, \mathsf{Accept})$.

Given an input $c_0 w_1 c_1 w_2 \ldots c_{j-2} w_{j-1} c_{j-1} w_j$ with $w_1, \ldots, w_j \in \Sigma_0^*$, $\overline{M}_j$ gradually compares the neighbouring factors w_i and w_{i+1} $(1 \leq i \leq j-1)$ from left

to right. First meta-instructions of type (1) are used repeatedly to compare w_1 to w_2, thereby deleting w_1 letter by letter from left to right, and encoding each letter u of w_2 by the letter d_u. Then meta-instructions of type (2) are used to compare w_2 to w_3, thereby deleting (the encoded version of) w_2 letter by letter from left to right, and encoding each letter u of w_3 by the letter d_u. This continues for indices $3, 4, \ldots, j-2$ until finally meta-instructions of type $(j-1)$ are used repeatedly to compare (the encoded version of) w_{j-1} and w_j, thereby deleting both syllables from left to right. It follows that each computation of $\overline{M}_j$ is 2-constrained, that is, $LC_j \in \mathcal{L}_I(2, \mathsf{FRR})$.

This example illustrates the fact that, considered as an input language of an FRR-automaton, a language may be classified essentially different (lower in the hierarchy) as when it is considered as a language of sentential forms. The reason for this behavior is the fact that the correctness preserving property need not hold for input languages. The following theorem summarizes these observations.

Theorem 2.
(a) $\mathcal{L}_{SF}(\mathsf{FRR}) \subset \mathcal{L}_I(\mathsf{FRR})$, *and*
(b) $\mathcal{L}_{SF}(i, \mathsf{FRR}) \subset \mathcal{L}_I(i, \mathsf{FRR})$ *for all* $i \in \mathbb{N}_+$.

Proof. For $i \geq 2$ the proper inclusion in (b) is an immediate consequence of Example 4, while for $i = 1$ this is simply part of the statement of Theorem 1. For proving (a) consider the language $L'_\infty := \{ (a^n b^n)^n \mid n \geq 1 \}$.

Claim 1. $L'_\infty \in \mathcal{L}_I(\mathsf{FRR})$.

Proof. We describe an FRR-automaton M' with tape alphabet $\Gamma := \Sigma_0 \cup \{A, B\}$ and input alphabet Σ_0 for the language L'_∞. Given an input of the form $w := a^{m_1} b^{n_1} a^{m_2} b^{n_2} \ldots a^{m_t} b^{n_t}$ $(t > 0, n_i, m_i > 0, i = 1, \ldots, t)$ M' will proceed as follows.

Each restarting configuration that M' reaches during its computation on input w will be of the form

$$C_j := q_0 ¢ A^{m_1 - j} B^{n_1 - j} \ldots A^{m_j - j} B^{n_j - j} a^{m_{j+1} - j} b^{n_{j+1} - j} \ldots a^{m_t - j} b^{n_t - j} \$,$$

where the initial configuration $q_0 ¢ w \$$ is just the special case C_0. M' will accept starting from C_j, if the tape content of C_j is from the regular language

$$¢ \cdot (AB)^* \cdot ab \cdot \$.$$

If the tape content is not of this form, then M' executes a cycle that comprises the following operations when starting from C_j:

- In each block of the form $A^{m_i - j} B^{n_i - j}$ $(1 \leq i \leq j)$, a factor AB is deleted. If $m_i - j = 1$ or $n_i - j = 1$, then M' halts and rejects.
- The first block from $a^+ b^+$, that is, $a^{m_{j+1} - j} b^{n_{j+1} - j}$, is rewritten into $A^{m_{j+1} - j - 1} B^{n_{j+1} - j - 1}$. If $m_{j+1} - j = 1$ or $n_{j+1} - j = 1$, then M' halts and rejects.

- In each block of the form $a^{m_{j+i}-j}b^{n_{j+i}-j}$ $(2 \leq i \leq t-j)$, a factor ab is deleted. Again if $m_{j+i} - j = 1$ of $n_{j+i} - j = 1$, then M' halts and rejects.
- Upon reaching the right delimiter \$, M' restarts.

If M' does not reject while performing these operations, then it reaches the restarting configuration C_{j+1}, that is, the computation of M' on input w has the form

$$C_0 \vdash^c_{M'} C_1 \vdash^c_{M'} \cdots \vdash^c_{M'} C_j,$$

and either M' halts and rejects starting from C_j, or $j = t-1$ and $C_j = q_0\text{¢}(AB)^{t-1}ab\$$, in which case M' halts and accepts. Thus, $L_I(M') = L'_\infty$. □

Claim 2. $L'_\infty \notin \mathcal{L}_{SF}(\mathsf{FRR})$.

Proof. Assume that $L'_\infty = L_{SF}(M)$ for some FRR-automaton M. Then M has tape alphabet Σ_0 only. Given an input of the form $(a^nb^n)^n$ for sufficiently large n, M cannot accept in a tail computation, that is, starting from the initial configuration $q_0\text{¢}(a^nb^n)^n\$$, an accepting computation of M will begin with a cycle that leads to a restarting configuration $q_0\text{¢}u\$$. As M must satisfy the Correctness Preserving Property, we see that $u \in L'_\infty$ holds. Further, as each rewrite step of M reduces the weight of the tape content with respect to some appropriate weight function, we see that u must be of the form $u = (a^mb^m)^m$ for some integer $m < n$. Thus, in the cycle above M deletes $m - n > 0$ blocks of the form a^nb^n, and it rewrites all remaining blocks into a^mb^m by deleting the factor $a^{n-m}b^{n-m}$. If n is sufficiently large, then M cannot ensure that the deleted blocks are of the correct length, that is, there exists an integer $j > 0$ and an index $n_1 \in \{1, \ldots, n\}$ such that M will also execute the cycle $(a^nb^n)^{n_1}a^{n+j}b^{n+j}(a^nb^n)^{n-n_1-1} \vdash^c_M (a^mb^m)^m$, which contradicts the Error Preserving Property. □

This completes the proof of Theorem 2. □

Next we introduce some notions in order to formalize the degree of word-order freedom of languages.

Definition 5. (a) *Two strings $u, v \in \Gamma^*$ are called* Parikh-equivalent, *denoted by $u \equiv v$, if $|u|_a = |v|_a$ holds for each $a \in \Gamma$.*

(b) *Let M be an* FRR*-automaton, and let $u, v \in L_{SF}(M)$. We say that u is* M-transformable *into v, denoted by $u \equiv\!>_M v$, if $u \equiv v$ holds, and for each string u_1 satisfying $u \vdash^c_M u_1$ and $|u| > |u_1|$, there exists a string v_1 such that $v \vdash^{c*}_M v_1$ and $u_1 \equiv v_1$.*

(c) *An* FRR*-automaton M is called* reduction-preserving *if, for each $i > 0$ and each $w \in L_{SF}(M, i+1)$, there exists a string $v \in L_{SF}(M, i)$ such that $u \equiv\!>_M v$.*

If M is reduction-preserving, then for each $i > 0$ and each string w that is accepted by M through an $(i+1)$-constrained computation, there exists a string v that is accepted by M through an i-constrained computation such that w and u are Parikh-equivalent, and even more, each string that is shorter than

w and that can be reached from w by a reduction modulo M is also Parikh-equivalent to some string that can be reached from v by a number of reductions modulo M. Intuitively, this means that by commuting w we obtain a string v that is of a lower degree of word-order complexity, but that admits essentially the same reductions (modulo commutation).

Example 5. The MRR-automaton M_1 of Example 2 is reduction-preserving. Indeed, let $n \geq 2$ and $u := a^n b^n a^n b^n \in L_{SF}(M_1, 2) \setminus L_{SF}(M_1, 1)$. Choose $v := a^{2n} b^{2n} \in L_{SF}(M_1, 1)$. Then $u \equiv_{M_1} v$, as $u \equiv v$ and $u \vdash^c_{M_1} u_1$ implies that $u_1 = a^{n-1} b^{n-1} a^{n-1} b^{n-1}$, and so $v_1 := a^{2n-2} b^{2n-2} \in L_{SF}(M_1, 1)$ satisfies the conditions $v \vdash^{c^*}_{M_1} v_1$ and $v_1 \equiv u_1$.

An FRR-automaton M is trivially reduction-preserving, if $L_{SF}(M, i) = L_{SF}(M, 1)$ holds for all $i \geq 1$. The following notion will be used to measure how far distant a reduction-preserving FRR-automaton is from this trivial case.

Definition 6. *A reduction-preserving* FRR*-automaton M is called* j-scalable *for some $j \in \mathbb{N}_+$ if, for each $i = 1, \ldots, j-1$, $L_{SF}(M, i+1) \notin \mathcal{L}_{SF}(i, \mathsf{FRR})$. A language L_{SF} of sentential forms is* j-scalable*, if there exists a j-scalable* FRR*-automaton M such that $L_{SF} = L_{SF}(M)$.*

Thus, if an FRR-automaton M is j-scalable, then, for all $1 \leq i \leq j-1$, the language $L_{SF}(M, i+1)$ of sentential forms that M accepts through $(i+1)$-constrained computations is not accepted by any FRR-automaton performing only i-constrained computations. Of course, each reduction-preserving FRR-automaton is 1-scalable. In the following the notion of scalability will serve as a synonym for the degree of word-order freedom of a language.

The following proposition is an immediate consequence of the above definition.

Proposition 1. *If an* FRR*-automaton M is j-scalable for some integer $j > 1$, then M is $(j-1)$-scalable as well.*

We now illustrate the above concepts by a few examples.

Example 6. As seen above the MRR-automaton M_1 of Example 2 is reduction-preserving. Further, the language $L_{SF}(M_1, 2) = \{ a^n b^n, a^n b^n a^n b^n \mid n > 0 \}$ does not coincide with the language of sentential forms that an FRR-automaton can accept by 1-constrained computations. Thus, M_1 is 2-scalable. However, it is not 3-scalable, as $L_{SF}(M_1) = L_{SF}(M_1, 2)$. On the other hand, the MRR-automaton M_2 of Example 3 is only 1-scalable, but it is not 2-scalable, as the language $L_{SF}(M_2, 2)$ is also accepted by the MRR-automaton M' performing only 1-constrained computations.

Example 7. The language LC_2 is easily seen to be 2-scalable, as it is accepted by the MRR-automaton MC_2 of Example 4 which performs only 2-constrained computations, and which can easily be modified to a reduction-preserving MRR-automaton. On the other hand, it is rather obvious that LC_2 does not coincide with the language $L_{SF}(M, 1)$ for any FRR-automaton M.

For $j = 3$ and $j = 4$, it is easily seen that whenever M is an FRR-automaton such that $LC_j = L_{SF}(M)$, then $L_{SF}(M, 1)$ is a finite subset of LC_j. Further, $L_{SF}(M, 2)$ is an infinite subset of LC_j that cannot be accepted by 1-constrained computations, while $L_{SF}(M, 3)$ cannot be accepted by 2-constrained computations. Also LC_4 cannot be accepted by 3-constrained computations. Thus, LC_3 is 3-scalable, and LC_4 is 4-scalable.

For $j \geq 5$, the language LC_j belongs to the class $\mathcal{L}_{SF}(j, \mathsf{FRR})$ (see Example 4), but it is not j-scalable. Indeed, if M is an FRR-automaton such that $LC_j = L_{SF}(M)$, then $L_{SF}(M, i)$ is finite for all $1 \leq i < \lceil j/2 \rceil$. Thus, for all $1 \leq i < \lceil j/2 \rceil - 1$, $L_{SF}(M, i+1) \in \mathcal{L}_{SF}(i, \mathsf{FRR})$, which implies that M is not j-scalable for $j \geq 5$.

Finally we turn to the family of languages LE_j $(j \geq 1)$.

Proposition 2. *For $j \geq 2$, LE_j is $(j-1)$-scalable, but it is not j-scalable.*

Proof. Let $j \geq 2$, and let M_j be the (deterministic) MRR-automaton with tape alphabet $\Delta_j = \{a_1, \ldots, a_j\}$ that is defined by the following meta-instructions, where π ranges over the set of permutations of the index set $\{1, 2, \ldots, j\}$, and i ranges over the set of indices $3, \ldots, j-1$:

$$
\begin{array}{ll}
(1.\pi) & (\mathfrak{c} \cdot a^*_{\pi(1)}, a_{\pi(1)}a_{\pi(2)} \to \lambda, \{a_{\pi(1)}, a_{\pi(2)}\}^*, a_{\pi(3)} \to \lambda, \\
 & \qquad \ldots, (\Delta_j \setminus \{a_{\pi(j)}\})^*, a_{\pi(j)} \to \lambda, \Delta_j^* \cdot \$); \\
(2.i) & (\mathfrak{c} \cdot a_2^*, a_2a_1 \to \lambda, a_1^*, a_3 \to \lambda, a_3^* \cdot a_1^*, a_4 \to \lambda, a_4^* \cdot a_1^*, \\
 & \qquad \ldots, a_i^* \cdot a_1^*, a_{i+1} \ldots a_j \to \lambda, (a_{i+1} \ldots a_j)^* \cdot \$); \\
(3) & (\mathfrak{c} \cdot \$, \mathsf{Accept})\,.
\end{array}
$$

Claim 1. $L_{SF}(M_j) = LE_j$, and every computation of M_j is $(j-1)$-constrained.

Proof. If the tape content is of the form

$$w(i) := a_2^n a_1^{m_1} a_3^n a_1^{m_2} a_4^n \ldots a_i^n a_1^{m_{i-1}} (a_{i+1} \ldots a_j)^n$$

for some integers $n, m_1, \ldots, m_{i-1} > 0$, then meta-instruction $(2.i)$ is applicable. It leads to acceptance if and only if $w(i) \in LE_j$, and the corresponding accepting computation is obviously i-constrained. If the tape content is not of the form above, then M_j can only apply the meta-instructions $(1.\pi)$, and hence, in each cycle M_j executes exactly $j-1$ delete operations. If we arrange the rewrite (delete) configurations of a computation of M_j into $j-1$ subsequences in such a way that the i-th delete configuration of each cycle is put into the i-th subsequence for all $i = 1, \ldots, j-1$, then it can be verified that the $j-1$ subsequences obtained are all monotone. It is not hard to see that M_j recognizes LE_j. □

It is easily seen that M_j is reduction-preserving. On the other hand, we have the following result, which completes the proof of Proposition 2.

Claim 2. For all $i \in \{2, \ldots, j-1\}$, $L_{SF}(M_j, i) \notin \mathcal{L}_{SF}(i-1, \mathsf{FRR})$.

Proof. Consider an input of the form $w(i)$ above such that $n = m_1 + \ldots + m_{i-1}$, and all integers $m_1, \ldots, m_{i-1}$ are sufficiently large. In order to transform $w(i)$ into a shorter sentence also belonging to $L_{SF}(M_j, i)$, at least one occurrence of each letter $a_1, a_2, \ldots, a_j$ must be deleted. If the integers $m_2, \ldots, m_{i-1}$ are larger than the size of the window of the automaton considered, then in order to delete at least one occurrence of each letter, the automaton must perform at least i delete operations. □

The next theorem, which is a consequence of the above proposition, shows that there exists an infinite sequence of classes of languages with an increasing degree of word-order freedom and word-order complexity as well.

Theorem 3. *For all* $i, j \in \mathbb{N}_+, i < j$,

(a) $\mathcal{L}_{SF}((j+1)\text{-scal-FRR}) \subset \mathcal{L}_{SF}(j\text{-scal-FRR})$.
(b) $\mathcal{L}_{SF}(i, j\text{-scal-FRR}) \subset \mathcal{L}_{SF}(i+1, j\text{-scal-FRR})$.

Proof. The inclusions in (a) follows from Proposition 1, while Proposition 2 shows that these inclusions are proper. As each $(i+1)$-constrained computation is also i-constrained, we obtain the inclusions in (b). Here Claim 2 of the proof of Proposition 2 implies that these inclusions are proper. □

4 Conclusions

We consider the notion of j-constrainability as a measure for the complexity of the word-order of (individual syntactic phenomena of) natural languages, while the notion of j-scalability serves as a measure for the magnitude of the word-order freedom of languages. We have seen that the word-order complexity can serve as a parameter for restarting automata. This fact is valuable particularly for the construction of syntactic analyzers for (complex) free word-order languages. Moreover, we have seen that the word-order complexity can be caused by phenomena that are based on fixed syntactic positions as well as by phenomena that are based on flexible (free) syntactic positions.

We have seen that each context-free language is the input language of a 1-constrained FRR-automaton, while the languages of sentential forms accepted by 1-constrained FRR-automata form a proper subclass of CFL (Theorem 1). On the other hand, we have seen that the classes of languages of sentential forms that are accepted by i-constrained FRR-automata form an infinite strict hierarchy with respect to the parameter i (Corollary 2). Do we obtain a corresponding hierarchy within CFL? Or is there an upper limit on the degree of constrainability for context-free languages of sentential forms of FRR-automata?

For example, consider the context-free language $L_{\text{pal}} := \{ ww^R \mid w \in \{a, b\}^* \}$. It is easily seen that this language is accepted by a 2-constrained FRR-automaton that, in each cycle, simply compares and removes the first symbol and the last symbol of the actual tape content. On the other hand, it appears highly unlikely that the language L_{pal} coincides with the language of sentential forms of

any 1-constrained FRR-automaton. Hence, we see that there seems to be context-free languages that belong to the difference set $\mathcal{L}_{SF}(2, \mathsf{FRR}) \setminus \mathcal{L}_{SF}(1, \mathsf{FRR})$. Are there also context-free languages in the set $\mathcal{L}_{SF}(i+1, \mathsf{FRR}) \setminus \mathcal{L}_{SF}(i, \mathsf{FRR})$ for larger values of i? Are there actually context-free languages that are not accepted as languages of sentential forms by any FRR-automaton at all?

For the future we also plan to characterize the *basic syntactic power* of tree-adjoining grammars [10] using techniques from [2], various types of categorial grammars (see, e.g., [1]) and of other tools [5,6] based on the notions outlined in this paper. This seems to be promising, in particular for grammars based on topological constraints [5], because of the similar (in fact topological) type of constraints studied here.

References

1. J. Baldridge and G.J. Kruijff. Multi-Modal Combinatory Categorial Grammar. In: *Proc. 10th Annual Meeting of the ACL*, Budapest.
2. M. Bodirsky, M. Kuhlmann, and M. Möhl. Well-nested drawings as models of syntactic structure. Accepted for publication at the 10th Conference on Formal Grammar and the Ninth Meeting on Mathematics of Language, Edinburgh, United Kingdom, 2005.
3. J. W. Bresnan, R. M. Kaplan, S. Peters, and A. Zaenen. Cross-serial dependencies in dutch. *Linguistic Inquiry*, 13:613–635, 1983.
4. N. Chomsky. *Syntactic Structures.* Mouton & Co., 1957, reprinted 1978.
5. R. Debusmann, D. Duchier, and J. Niehren. The XDG Grammar Development Kit: In: *Proceedings of the MOZ04 Conference* Charleroi/BEL, *Lecture Notes in Computer Science* 3389, Springer, 2004, 190–201.
6. R. Gramatovici and C. Martin-Vide. Contextual Grammars and Dependency Trees. In: B. Daille and E. Morin (eds.), *Actes de TALN. 10e Conference sur le Traitement Automatique des Langues Naturelles, TALN 2003*, Batz-sur-Mer, France, 2003, 135–144.
7. H. Haider. Argument structure: Semantic basis and syntactic effects. Class notes from the Third European Summer School in Language, Logic and Information, Saarbrücken, 1991.
8. T. Holan, V. Kuboň, K. Oliva, and M. Plátek. Two Useful Measures of Word Order Complexity. In: A. Polguere and S. Kahane (eds.), *Proc. of the Coling '98 Workshop "Processing of Dependency-Based Grammars"*, University of Montreal, 1998, 21–28.
9. P. Jančar, F. Mráz, M. Plátek, and J. Vogel. On Monotonic Automata with a Restart Operation, *J. Autom. Lang. Comb.* 4 (1999) 283-292.
10. A.K. Joshi and Y. Shabes. Tree-Adjoining Grammars. In: G. Rosenberg and A. Salomaa (eds.), *Handbook of Formal Languages*, Vol. 3, Berlin, Springer, 1997, 69–123.
11. T. Jurdziński and F. Otto, On left-monotone restarting automata. *Mathematische Schriften Kassel* 17/03, Universität Kassel, 2003.
12. T. Jurdziński and F. Otto, Shrinking restarting automata. In: J. Jedrzejowicz and A. Szepietowski (eds.), *MFCS 2005, Proc., Lecture Notes in Computer Science* , Springer, Berlin, 2005, to appear.
13. K. Oliva, P. Květoň, and R. Ondruška. The computational complexity of rule-based part-of-speech tagging. In V. Matousek and P. Mautner (eds.), *TSD 2003,*

Proc., *Lecture Notes in Computer Science* 2807, pages 82–89. Springer, Berlin, 2003.

14. F. Otto. Restarting automata and their relations to the Chomsky hierarchy. In: Z. Ésik and Z. Fülöp (eds.), *Developments in Language Theory, Proc. DLT'2003*, *Lecture Notes in Computer Science* 2710, Springer, Berlin, 2003, 55–74.
15. M. Plátek, M. Lopatková, and K. Oliva. Restarting automata: motivations and applications. In: M. Holzer (ed.), *Workshop 'Petrinetze' and 13. Theorietag 'Formale Sprachen und Automaten', Proc.*, Institut für Informatik, Technische Universität München, 2003, 90-96.
16. M. Plátek, F. Otto, and F. Mráz. Restarting automata and variants of j-monotonicity. In E. Csuhaj-Varjú, C. Kintala, D. Wotschke, and G. Vaszil, editors, *Descriptional Complexity of Formal Systems, Proceedings DCFS 2003* , pages 303–312. MTA SZTAKI, Budapest, 2003.
17. O. Rambow and A. Joshi. A processing model for free word order languages. In J. C. Clifton, L. Frazier, and K. Rayner, editors, *Perspectives on sentence processing*, pages 267–301. Lawrence Erlbaum Associates, 1994.
18. S. Shieber. Evidence against the context-freeness of natural language. *Linguistics and Philosophy*, 8:333–343, 1985.
19. M. Steedman. Linguistic and Computational Theories of Grammar. Course at University of Edinburgh, 2005, http://www.iccs.informatics.ed.ac.uk/ steedman/tl/home.html