International Journal of Foundations of Computer Science

Jumping Grammars

ZBYNĚK KŘIVKA

IT4Innovations Centre of Excellence, Department of Information Systems, Faculty of Information Technology, Brno University of Technology, Božetěchova 2, Brno 612 66, Czech republic
krivka@fit.vutbr.cz
<http://www.fit.vutbr.cz/~krivka>

ALEXANDER MEDUNA

IT4Innovations Centre of Excellence, Department of Information Systems, Faculty of Information Technology, Brno University of Technology, Božetěchova 2, Brno 612 66, Czech republic
meduna@fit.vutbr.cz

Received (Day Month Year)
Communicated by (xxxxxxxxxx)

This paper introduces and studies *jumping grammars*, which represent a grammatical counterpart to the recently introduced jumping automata. These grammars are conceptualized just like classical grammars except that during the applications of their productions, they can jump over symbols in either direction within the rewritten strings. More precisely, a jumping grammar rewrites a string z according to a rule $x \rightarrow y$ in such a way that it selects an occurrence of x in z, erases it, and inserts y anywhere in the rewritten string, so this insertion may occur at a different position than the erasure of x.

The paper concentrates its attention on investigating the generative power of jumping grammars. More specifically, it compares this power with that of jumping automata and that of classical grammars. A special attention is paid to various context-free versions of jumping grammars, such as regular, right-linear, linear, and context-free grammars of finite index. In addition, we study the semilinearity of context-free, context-sensitive, and monotonous jumping grammars. We also demonstrate that the general versions of jumping grammars characterize the family of recursively enumerable languages. In its conclusion, the paper formulates several open problems and suggests future investigation areas.

Keywords: Modified grammars; discontinuous rewriting; generative power; jumping finite automata; semilinearity; finite index.

1991 Mathematics Subject Classification: 68Q05, 68Q10, 68Q42, 68Q45, 68Q70

1. Introduction

The introduction of this paper consists of four subsections. Section 1.1 explains the reason why this paper modifies classical grammars. Section 1.2 informally describes this modification and important results. Section 1.3 sketches a straightforward application of this new concept in DNA computing. Section 1.4 gives the contents of the rest of the paper.

1.1. *Motivation*

Processing information in a largely discontinuous way represents a common computational phenomenon today [1, 2, 7]. Indeed, consider a process p that deals with information i. During a single computational step, p can read a piece of information x in i, erase it, generate a new piece of information y, and insert y into i possibly far away from the original occurrence of x, which was erased. Therefore, intuitively speaking, during its computation, p keeps jumping across i as a whole. To explore computation like this systematically and rigorously, computer science obviously needs formal models that reflect it in an adequate way.

Traditionally, formal language theory has always provided computer science with language-defining models to explore various information processors mathematically, so it should do so for the purpose sketched above, too. However, the classical versions of these models, such as automata and grammars, work on words strictly continuously, and as such, they can hardly serve as appropriate models of this kind. Therefore, a proper formalization of processors that work in the way described above necessities an adaptation of classical automata and grammars so they work on words discontinuously. At the same time, any adaptation of this kind should conceptually maintain the original structure of these models as much as possible so computer science can quite naturally base its investigation upon these newly adapted models by analogy with the standard approach based upon their classical versions. Simply put, these new models should work on words in a discontinuous way while keeping their structural conceptualization unchanged.

As a matter of fact, finite automata, which definitely represent important language-accepting models of computation, have been modified in this way recently. Indeed, these modified finite automata, referred to as *jumping finite automata* (see [9]), work just like classical finite automata except that they read input words discontinuously—that is, it does not read the input string in a symbol-by-symbol left-to-right way. Instead, after reading a symbol, they can jump over some symbols within the words and continue their computation from there.

1.2. *Grammatical Approach*

As already pointed out, apart from automata, there exist grammars as the other fundamental language-defining models in formal language theory. As a result, the recent introduction of jumping automata obviously leads to introducing jumping grammars—the subject of this paper.

Recall that the notion of a classical grammar G represents a language-generating rewriting system based upon an alphabet of symbols and a finite set of productions. The alphabet of symbols is divided into two disjoint subalphabets—the alphabet of terminal symbols and the alphabet of nonterminal symbols. Each production rule represents a pair of the form (x, y), where x and y are strings over the alphabet of G. Customarily, (x, y) is written as $x \to y$, where x and y are referred to as the left-hand side and the right-hand side of $x \to y$. Starting from a special start nonterminal symbol, G repeatedly rewrites strings according to its productions until it obtains a sentence—that is, a string that solely con-

sists of terminal symbols; the set of all sentences represents the language generated by the grammar. In a greater detail, G rewrites a string z according to $x \to y$ so it (1) selects an occurrence of x in z, (2) erases it, and (3) inserts y precisely at the position of this erasure. More formally, let $z = uxv$, where u and v are strings. By using $x \to y$, G rewrites uxv as uyv.

The notion of a *jumping grammar*—that is, the key notion introduced in this paper—is conceptualized just like that of a classical grammar; however, it rewrites strings in a slightly different way. Consider G, described above, as a jumping grammar. Let z and $x \to y$ have the same meaning as above. G rewrites a string z according to $x \to y$ so it performs (1) and (2) as described above, but during (3), G can jump over a portion of the rewritten string in either direction and inserts y there. More formally, by using $x \to y$, G rewrites ucv as udv, where u, v, w, c, d are strings such that either (i) $c = xw$ and $d = wy$ or (ii) $c = wx$ and $d = yw$. Otherwise, it coincides with the standard notion of a grammar.

Regarding the general versions of jumping grammars, we demonstrate that they are as powerful as classical general grammars. As there exist many important special versions of these classical grammars, we discuss their jumping counterparts in the present paper as well. Perhaps most importantly, we study the jumping versions of context-free grammars and their special cases, including regular grammars, right-linear grammars, linear grammars, and context-free grammars of finite index. Surprisingly, all of them have a different power than their classical counterparts.

As already pointed out, the fundamental purpose of this paper is to demonstrate the discussion of jumping grammars as a significant and up-to-date topic of modern formal language theory, and in this sense, the present paper is primarily meant as a proposal of a new investigation area in this theory. Of course, it cannot present a systematic and compact body of rigorous knowledge that exhaustively and thoroughly represents the theory of jumping grammars. Instead, it narrows its investigation to a key topic concerning them. More specifically, since the determination of the power of language-defining devices has always fulfilled a central role in formal language theory, this paper pays its principle attention to the study of the generative power of jumping grammars. First, it compares this power with the power of jumping finite automata. More specifically, it demonstrates that regular jumping grammars are as powerful as jumping finite automata. Furthermore, right-linear jumping grammars are equivalent with the generalized jumping finite automata. The paper also compares the language families resulting from jumping versions of grammars with those resulting from standard versions of the corresponding grammars and demonstrates that most of them differ; perhaps most surprisingly, this difference concerns the families contained in the well-known Chomsky hierarchy.

1.3. *Applications Perspectives*

As follows from the beginning of this section, jumping grammars serve as grammatical models that allow us to explore information processing performed in a discontinuous way adequately and rigorously. Consequently, applications of these grammars are expected in any scientific area involving this kind of information processing, ranging from applied

mathematics through computational linguistics up to biology. Taking into account the way these grammars are conceptualized, we see that they are particularly useful and applicable under the circumstances that primarily concern the number of occurrences of various symbols or substrings rather than their mutual context. To give a more specific illustration, consider DNA computing, whose significance is indisputable in computer science at present. Recall that a DNA is a molecule encoding genetic information by a repetition of four basic units called nucleotides—namely, guanine, adenine, thymine, and cytosine, denoted by letters G, A, T, and C, respectively. In terms of formal language theory, a DNA is described as a string over $\{G, A, T, C\}$; for instance,

$$GGGGAGTGGGATTGGGAGAGGGGTTTGCCCCGCTCCC$$

represents a string like this. Suppose that a DNA-computing-related investigation needs to study all the strings that contain the same number of Cs and Gs and the same number of As and Ts; for instance, $CGGCATCCGGTA$ is a proper string like this, but $CGCACCGGTA$ is not. Consider the jumping right-linear grammar containing productions

$$1 \to C2,\ 2 \to G1,\ 1 \to 3,\ 3 \to A4,\ 4 \to T3,\ 3 \to \varepsilon$$

where 1 through 4 are nonterminal symbols with 1 being the start nonterminal, ε is the empty string, and G, A, T, and C are, of course, terminal symbols. As obvious, this grammar generates the language consisting of all the strings satisfying the above-stated requirements. Therefore, as this trivial example illustrates, jumping grammars may fulfill a useful role studies related to DNA computing in the future.

1.4. *Organization*

The rest of the paper is organized as follows. Section 2 recalls all the terminology needed in this paper and introduces a variety of jumping grammars including some examples. Section 3 presents fundamental results achieved in this paper. Section 4 closes all the study by pointing out important open problem areas.

2. Preliminaries and Definitions

For an alphabet, V, V^* represents the free monoid generated by V under the operation of concatenation. The unit of V^* is denoted by ε. Set $V^+ = V^* - \{\varepsilon\}$. For $w \in V^*$, $|w|$ and $\text{alph}(w)$ denote the *length* of w and the set of symbols occurring in w, respectively. For all $i \geq 0$, the *ith power of* w, denoted by w^i, is recursively defined as (1) $w^0 = \varepsilon$, and (2) $w^i = ww^{i-1}$, for $i \geq 1$. Let $N \subseteq V$; $\text{occur}(N, w)$ denotes the number of occurrences of symbols from N in w. Let $w = a_1a_2 \cdots a_n$, where $a_i \in V$ for all $i = 1, 2, \ldots, n$, for some $n \geq 0$ ($x = \varepsilon$ iff $n = 0$). The *set of all permutations* of w, $\text{perm}(w)$, is defined as $\text{perm}(w) = \{b_1b_2 \cdots b_n \mid b_i \in \text{alph}(w) \text{ for all } i = 1, 2, \ldots, n, \text{ and } (b_1, b_2, \ldots, b_n) \text{ is a permutation of } (a_1, a_2, \ldots, a_n)\}$. Let $L \subseteq V^*$, $\text{perm}(L) = \{w \mid w \in \text{perm}(w'), w' \in L\}$. Let X and Y be sets; we call X and Y to be *incomparable* if $X \not\subseteq Y$ and $Y \not\subseteq X$.

Definition 1. *A* general grammar *(*GG *for short) is a quadruple, $G = (V, T, P, S)$, where V is an alphabet, $T \subseteq V$, P is a finite relation from $V^* - T^*$ to V^*, and $S \in V - T$ is the start nonterminal. Members of T and $(V - T)$ are called terminals and nonterminals, respectively. Let $N = V - T$ hereafter. Members of P are called* productions*; instead of $(x, y) \in P$, we write $x \to y$ throughout the paper. For brevity, we sometimes denote a production $x \to y$ with a unique label p as $p\colon x \to y$, and instead of $x \to y \in P$, we simply write $p \in P$. For $p \in P$, $lhs(p)$ and $rhs(p)$ denotes the left-hand side x and the right-hand side y of production p, respectively.*

Let r be a relation over V^*. As usual, for every $n \geq 0$, the n-fold product of r is denoted by r^n. The transitive-reflexive closure and the transitive closure of r are denoted by r^* and r^+, respectively. Let k be a positive integer. Let $r_k = \{(x, y) \mid (x, y) \in r, occur(N, x) \leq k, occur(N, y) \leq k\}$. Unless explicitly stated otherwise, we write xry instead of $(x, y) \in r$ in what follows.

Let $G = (V, T, P, S)$ be a GG and r be a relation over V^*. Set

$$L(G, r) = \{x \in T^* \mid Sr^*x\}.$$

$L(G, r)$ is said to be the language that G generates by using r. *For any $X \subseteq \Gamma_{GG}$ (as defined below, Γ_{GG} denotes the set of GGs) and any set R of relations over V^*, set*

$$\mathscr{L}(X, R) = \{L(G, r) \mid G \in X, r \in R\}.$$

If R contains only one relation r, we simplify $\mathscr{L}(X, \{r\})$ to $\mathscr{L}(X, r)$ for brevity.

Next, we introduce four modes of jumping derivation steps *as relations over V^*. Formally, for $u, v \in V^*$, we define these relations as follows*

(i) $u \ {}_s\Rightarrow v$ *in G iff there exist $x \to y \in P$ and $w, z \in V^*$ such that $u = wxz$ and $v = wyz$;*
(ii) $u \ {}_{lj}\Rightarrow v$ *in G iff there exist $x \to y \in P$ and $w, t, z \in V^*$ such that $u = wtxz$ and $v = wytz$;*
(iii) $u \ {}_{rj}\Rightarrow v$ *in G iff there exist $x \to y \in P$ and $w, t, z \in V^*$ such that $u = wxtz$ and $v = wtyz$;*
(iv) $u \ {}_j\Rightarrow v$ *in G iff* $u \ {}_{lj}\Rightarrow v$ *or* $u \ {}_{rj}\Rightarrow v$ *in G.*

Sometimes, we need to explicitly specify the sequence of productions applied during some of these four jumping derivation modes. Consider (i). To express that a GG G applies a production p during $u \ {}_s\Rightarrow v$, we write $u \ {}_s\Rightarrow v \ [p]$, where $p \in P$. By $u \ {}_s\Rightarrow^* v \ [\pi]$, where π is a sequence of productions from P, we express that G makes $u \ {}_s\Rightarrow^* v$ by using π. Analogously, we present this specification in terms of the other three modes (ii) through (iv)—${}_{lj}\Rightarrow$, ${}_{rj}\Rightarrow$, and ${}_j\Rightarrow$.

Next, we define several special cases of GGs. Let G be a GG.

- G is a *monotonous grammar* (*MONG* for short) if every $x \to y \in P$ satisfies $|x| \leq |y|$.
- G is a *context-sensitive grammar* (*CSG* for short) if every $x \to y \in P$ satisfies $x = \alpha A \beta$ and $y = \alpha\gamma\beta$ such that $A \in N$, $\alpha, \beta \in V^*$, and $\gamma \in V^+$.
- G is a *context-free grammar* (*CFG* for short) if every $x \to y \in P$ satisfies $x \in N$.
- G is an *ε-free context-free grammar* (*CFG*$^{-\varepsilon}$ for short) if G is a CFG and every $x \to y \in P$ satisfies $y \neq \varepsilon$.

- G is a *linear grammar* (*LG* for short) if G is a CFG and every $x \to y \in P$ satisfies $y \in T^*NT^* \cup T^*$.
- G is a *right-linear grammar* (*RLG* for short) if G is a CFG and every $x \to y \in P$ satisfies $y \in T^*N \cup T^*$.
- G is a *regular grammar* (*RG* for short) if G is a CFG and every $x \to y \in P$ satisfies $y \in TN \cup T$.

Let Γ_{GG}, Γ_{MONG}, Γ_{CSG}, Γ_{CFG}, $\Gamma_{CFG^{-\varepsilon}}$, Γ_{LG}, Γ_{RLG}, and Γ_{RG} denote the sets of all GGs, MONGs, CSGs, CFGs, CFG$^{-\varepsilon}$s, LGs, RLGs, and RGs, respectively.

To illustrate the above-introduced notation, let $G = (V, T, P, S)$ be a RLG; then, $L(G, {}_j\Rightarrow) = \{x \in T^* \mid S \, {}_j\Rightarrow^* x\}$, and $\mathscr{L}(\Gamma_{RLG}, {}_j\Rightarrow) = \{L(G, {}_j\Rightarrow) \mid G \in \Gamma_{RLG}\}$. To give another example, $\mathscr{L}(\Gamma_{CFG}, {}_s\Rightarrow)$ denotes the family of all context-free languages.

Definition 2. *Set* $\mathbf{REG} = \mathscr{L}(\Gamma_{RLG}, {}_s\Rightarrow)$, $\mathbf{LIN} = \mathscr{L}(\Gamma_{LG}, {}_s\Rightarrow)$, $\mathbf{CF} = \mathscr{L}(\Gamma_{CFG}, {}_s\Rightarrow)$, $\mathbf{CS} = \mathscr{L}(\Gamma_{MONG}, {}_s\Rightarrow)$, *and* $\mathbf{RE} = \mathscr{L}(\Gamma_{GG}, {}_s\Rightarrow)$. *Let k be a positive integer. Set* $\mathbf{CF}_k = \mathscr{L}(\Gamma_{CFG}, \{{}_s\Rightarrow_i \mid 1 \leq i \leq k\})$ *and* $\mathbf{CF}_{fin} = \{L \mid L \in \mathbf{CF}_i$, for some $i \geq 1\}$. *For further details concerning finite index of grammars, see Chapter 3 in* [3]. *Let* **FIN**, **REG**, **LIN**, **CF**, **CS**, *and* **RE** *denote the family of finite, regular, linear, context-free, context-sensitive, and recursively enumerable languages, respectively.*

Recall (see [10])

$$\mathbf{FIN} \subset \mathbf{REG} \subset \mathbf{LIN} \subset \mathbf{CF}_{fin} \subset \mathbf{CF} \subset \mathbf{CS} \subset \mathbf{RE}$$

Example 3. *Consider the following RG*

$$G = (\{A, B, C, a, b, c\}, \Sigma = \{a, b, c\}, P, A)$$

where $P = \{A \to aB, B \to bC, C \to cA, C \to c\}$.

$$L(G, {}_s\Rightarrow) = \{abc\}\{abc\}^*$$

$$L(G, {}_j\Rightarrow) = \{w \in \Sigma^* \mid occur(\{a\}, w) = occur(\{b\}, w) = occur(\{c\}, w)\}$$

Notice that although $L(G, {}_s\Rightarrow)$ *is regular,* $L(G, {}_j\Rightarrow) \in \mathbf{CS}$ *is a well-known non-context-free language.*

Example 4. *Consider the following CSG* $G = (\{S, A, B, a, b\}, \{a, b\}, P, S)$ *containing the following productions*

$$\begin{array}{rcl} S & \to & aABb \\ S & \to & ab \\ AB & \to & AABB \\ aA & \to & aa \\ Bb & \to & bb \end{array}$$

Trivially, $L(G, {}_s\Rightarrow) = \{a^n b^n \mid n \geq 1\}$. *Using* ${}_j\Rightarrow$, *we can make the following derivation sequence (the rewritten substring is underlined):*

$$\underline{S} \; {}_j\Rightarrow a\underline{AB}b \; {}_j\Rightarrow \underline{aA}AB\underline{Bb} \; {}_j\Rightarrow^2 aa\underline{aA}Bbb \; {}_j\Rightarrow a\underline{Bb}baa \; {}_j\Rightarrow abbbaa$$

Notice that $L(G, {}_s\Rightarrow)$ is context-free, but we cannot generate this language by any CFG, CSG or even MONG in jumping derivation mode.

Lemma 5. $\{a\}^*\{b\}^* \notin \mathscr{L}(\Gamma_{MONG}, {}_j\Rightarrow)$.

Proof. Assume that there exists a MONG $G = (V, T, P, S)$ such that $L(G, {}_j\Rightarrow) = \{a\}^*\{b\}^*$. Let $p\colon\ x \to y \in P$ be the last applied production during a derivation $S\ {}_j\Rightarrow^+ w$ where $w \in L(G, {}_j\Rightarrow)$; that is, $S\ {}_j\Rightarrow^* uxv\ {}_j\Rightarrow w\ [p]$ where $u, v, w \in T^*$ and $y \in \{a\}^+ \cup \{b\}^+ \cup \{a\}^+\{b\}^+$. In addition, assume that the sentential form uxv is longer than x such that $uv \in \{a\}^+\{b\}^+$.

(i) If y contains at least one symbol b, the last jumping derivation step can place y at the beginning of the sentence and create a string from $\{a, b\}^*\{b\}\{a, b\}^*\{a\}\{a, b\}^*$ that does not belong to $\{a\}^*\{b\}^*$.
(ii) By analogy, if y contains at least one symbol a, the last jumping derivation step can place y at the end of the sentence and therefore, place at least one a behind some bs.

This is a contradiction, so there is no MONG that generates regular language $\{a\}^*\{b\}^*$ using ${}_j\Rightarrow$. □

We re-open a discussion related to Lemma 5 in the last section of this paper.

Corollary 6. *The following pairs of language families are incomparable, but not disjoint:*

(i) **REG** *and* $\mathscr{L}(\Gamma_{MONG}, {}_j\Rightarrow)$*;*
(ii) **CF** *and* $\mathscr{L}(\Gamma_{MONG}, {}_j\Rightarrow)$*;*
(iii) **REG** *and* $\mathscr{L}(\Gamma_{RG}, {}_j\Rightarrow)$*;*
(iv) **CF** *and* $\mathscr{L}(\Gamma_{RG}, {}_j\Rightarrow)$*.*

Proof. Since $\mathbf{REG} \subset \mathbf{CF}$, it is sufficient to prove that $\mathbf{REG} - \mathscr{L}(\Gamma_{MONG}, {}_j\Rightarrow)$, $\mathscr{L}(\Gamma_{RG}, {}_j\Rightarrow) - \mathbf{CF}$, and $\mathbf{REG} \cap \mathscr{L}(\Gamma_{RG}, {}_j\Rightarrow)$ are non-empty. By Lemma 5, $\{a\}^*\{b\}^* \in \mathbf{REG} - \mathscr{L}(\Gamma_{MONG}, {}_j\Rightarrow)$. In Example 3, we define a jumping RG that generates a non-context-free language that belongs to $\mathscr{L}(\Gamma_{RG}, {}_j\Rightarrow) - \mathbf{CF}$. Observe that regular language $\{a\}^*$ belongs to $\mathscr{L}(\Gamma_{RG}, {}_j\Rightarrow)$, so $\mathbf{REG} \cap \mathscr{L}(\Gamma_{RG}, {}_j\Rightarrow)$ is non-empty. □

As even some very simple regular language such as $\{a\}^+\{b\}^+$ cannot be generated by jumping derivation in CSGs or even MONGs, we pinpoint the following open problem and state a theorem comparing these families with context-sensitive languages.

Open Problem 7. *Are* $\mathscr{L}(\Gamma_{CFG}, {}_j\Rightarrow) \subseteq \mathscr{L}(\Gamma_{CSG}, {}_j\Rightarrow)$ *and* $\mathscr{L}(\Gamma_{CSG}, {}_j\Rightarrow) \subseteq \mathscr{L}(\Gamma_{MONG}, {}_j\Rightarrow)$ *proper?*

Theorem 8. $\mathscr{L}(\Gamma_{MONG}, {}_j\Rightarrow) \subset \mathbf{CS}$.

Proof. Clearly, a jumping MONG can be simulated by some linear bounded automata, so $\mathscr{L}(\Gamma_{MONG}, {}_j\Rightarrow) \subseteq \mathbf{CS}$. That is, by Lemma 5 this theorem holds. □

Example 9. *Assume a language with arithmetic expressions with three kinds of brackets and omitted operator symbols, literals, and identifiers. This bracket language is generated by classical CFG* $G = (\{E, (,), \{, \}, [,]\}, \{(,), \{, \}, [,]\}, \{E \rightarrow (E)E,\ E \rightarrow \{E\}E,\ E \rightarrow [E]E,\ E \rightarrow \varepsilon\},\ E)$. *G has no finite index. Consider jumping RLG H with the same start nonterminal, set of nonterminals and terminals as in G. The set of productions in H contains*

$$\begin{array}{lcllcl} E & \rightarrow & ()E & \qquad & E & \rightarrow & \{\}E \\ E & \rightarrow & []E & & E & \rightarrow & \varepsilon \end{array}$$

Observe that $L(G, {}_s\Rightarrow) = L(H, {}_j\Rightarrow)$ *and, in addition, H is of index* 1.

Definition 10. *[9] A* general jumping finite automaton*, a GJFA for short, is a quintuple* $M = (Q, \Sigma, R, s, F)$*, where Q is finite set of* states*,* Σ *is the* input alphabet*,* $Q \cap \Sigma = \emptyset$*,* $R \subseteq Q \times \Sigma^* \times Q$ *is finite,* $s \in Q$ *is the* start state*, 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* $py \rightarrow q \in R$*. If* $py \rightarrow q \in R$ *implies that* $|y| \leq 1$*, then M is a* jumping finite automaton*, a JFA for short.*

A configuration *of M is any string in* $\Sigma^*Q\Sigma^*$*. The binary* jumping relation*, symbolically denoted by* $\curvearrowright$*, over* $\Sigma^*Q\Sigma^*$*, is defined as follows. Let* x*,* z*,* x'*,* $z' \in \Sigma^*$ *such that* $xz = x'z'$ *and* $py \rightarrow q \in R$*; then, M makes a* jump *from* $xpyz$ *to* $x'qz'$*, symbolically written as* $xpyz \curvearrowright x'qz'$*. In the standard manner, we extent* $\curvearrowright$ *to* $\curvearrowright^m$*, where* $m \geq 0$*,* $\curvearrowright^+$*, and* $\curvearrowright^*$*.*

The language *accepted by M, denoted by* $L(M)$*, is defined as* $L(M) = \{uv \mid u, v \in \Sigma^*,\ usv \curvearrowright^* f,\ f \in F\}$.

GJFA *and* **JFA** *denote the families of languages accepted by GJFAs and JFAs, respectively.*

Recall that $\mathbf{JFA} \subset \mathbf{GJFA}$, $\mathbf{FIN} \subset \mathbf{GJFA}$, and $\mathbf{FIN}$ and $\mathbf{JFA}$ are incomparable, respectively (see Theorems 16, 18, and 23 in [9]).

3. Results

In this section, the generative power of several kinds of jumping grammars is discussed and the main results are demonstrated.

3.1. *Relations between Jumping Automata and Jumping Grammars*

We demonstrate that the generative power of regular and right-linear jumping grammars is the same as accepting power of jumping finite automata and general jumping finite automata, respectively. Consequently, the following equivalence and results from [9] imply several additional properties of languages that are generated by regular and right-linear jumping grammars such as closure properties and decidability.

Lemma 11. $\mathbf{GJFA} \subseteq \mathscr{L}(\Gamma_{RLG}, {}_j\Rightarrow)$.

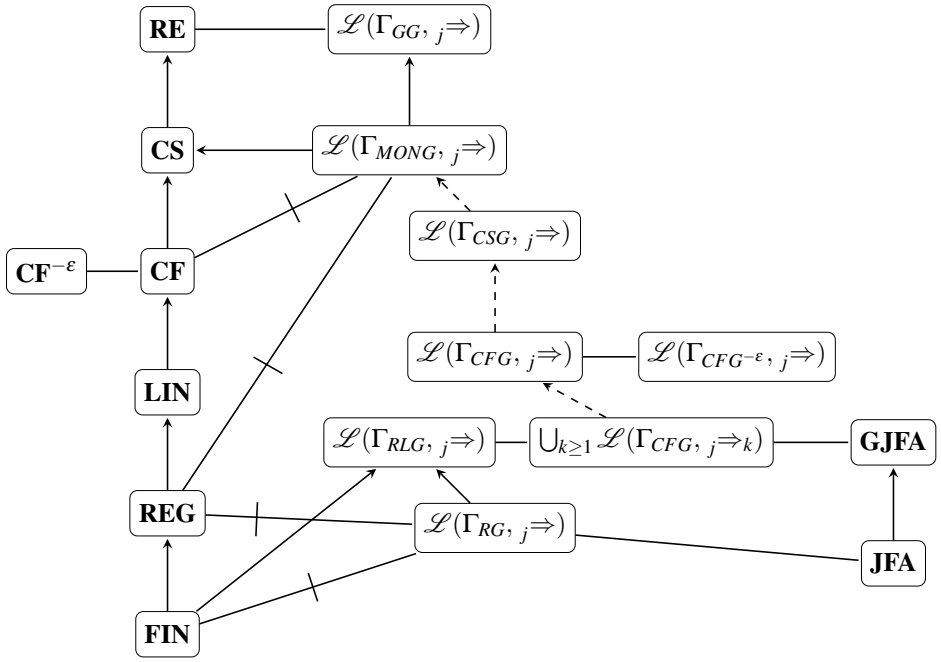

Fig. 1. A hierarchy of language families closely related to the language families resulting from jumping grammars is shown. If there is a line or an arrow from family X to family Y in the figure, then $X = Y$ or $X \subset Y$, respectively. If there is a dashed arrow from X to Y, then $X \subseteq Y$, but $X \subset Y$ represents an open problem. A crossed line represents the incomparability between connected families. (It is noteworthy that the figure describes only some of the language-family relations that are crucially important in terms of the present paper; however, by no means, it gives an exhaustive description of these relations.)

Proof.

Construction. For every GJFA $M = (Q, \Sigma, R, s, F)$, we construct a RLG $G = (Q \cup \Sigma \cup \{S\}, \Sigma, P, S)$, where S is a new nonterminal, $S \notin Q \cup \Sigma$, such that $L(M) = L(G, {}_j\Rightarrow)$.

Set $P = \{S \to f \mid f \in F\} \cup \{q \to xp \mid px \to q \in R\} \cup \{q \to x \mid sx \to q \in R\}$.

Basic Idea. The principle of the conversion is analogical to the conversion from classical lazy finite automata to equivalent RLGs with sequential derivation mode (see Section 2.6.2 in [11] and Theorem 4.1 in [10]).

The states of M are used as nonterminals in G. In addition, we introduce new start nonterminal S in G. The input symbols Σ are terminal symbols in G.

During the simulation of M in G there is always exactly one nonterminal symbol in the sentential form until the last jumping derivation step that produces the string of terminal symbols. If there is a sequence of jumping moves $sw \curvearrowright^* pyxy' \curvearrowright qyy' \curvearrowright^* f$ in M, then G simulates it by jumping derivation $S \;{}_j\Rightarrow f \;{}_j\Rightarrow^* zqz' \;{}_j\Rightarrow yxpy' \;{}_j\Rightarrow^* w$, where $yy' = zz'$. Firstly, S is nondeterministically rewritten to some f in G to simulate the en-

trance to the corresponding accepting final state of M. Then, for each rule $px \to q$ in M that processes substring x in the input string, there is x generated by the corresponding production of the form $q \to xp$ in G. As the last jumping derivation step in G, we simulate the first jumping move of M from the start state s by rewriting the only nonterminal in the sentential form of G to a string of terminals and the simulation of M by G is completed. □

Lemma 12. $\mathscr{L}(\Gamma_{RLG}, {}_j\Rightarrow) \subseteq$ **GJFA**.

Proof.
Construction. For every RLG $G = (V, T, P, S)$, we construct a GJFA $M = (N \cup \{\sigma\}, T, R, \sigma, \{S\})$, where σ is a new start state, $\sigma \notin V$ and $N = V - T$, such that $L(G, {}_j\Rightarrow) = L(M)$. Set $R = \{Bx \to A \mid A \to xB \in P, A, B \in N, x \in T^*\} \cup \{\sigma x \to A \mid A \to x \in P, x \in T^*\}$.

Basic Idea. In the simulation of G in M we use nonterminals N as states, new state σ as the start state, and terminals T corresponds to input symbols of M. In addition, the start nonterminal of G corresponds to the only final state of M. Every application of a production from P in G is simulated by a move according to the corresponding rule from R constructed above. If there is a jumping derivation $S \; {}_j\Rightarrow^* yAy' \; {}_j\Rightarrow zxBz' \; {}_j\Rightarrow^* w$ in G, then M simulates by jumping moves $\sigma w \curvearrowright^* Bzxz' \curvearrowright Ayy' \curvearrowright^* S$, where $yy' = zz'$. □

Theorem 13. **GJFA** $= \mathscr{L}(\Gamma_{RLG}, {}_j\Rightarrow)$.

Proof. This theorem holds by Lemma 11 and 12. □

In the following theorem, consider jumping finite automata that processes only one input symbol in one move. We state their equivalence with jumping RGs.

Theorem 14. **JFA** = $\mathscr{L}(\Gamma_{RG}, {}_j\Rightarrow)$.

Proof. Prove this statement by analogy with the proof of Theorem 13 with $x \in T$. □

3.2. *Relations between the Language Families Resulting from Various Jumping Grammars*

As its title indicates, the present section establishes results concerning relations between language families generated by jumping versions of grammars introduced earlier in this paper.

Theorem 15. $\mathscr{L}(\Gamma_{RLG}, {}_j\Rightarrow) = \mathscr{L}(\Gamma_{LG}, {}_j\Rightarrow) = \bigcup_{k\geq 1} \mathscr{L}(\Gamma_{CFG}, {}_j\Rightarrow_k)$.

Proof. Since $\mathscr{L}(\Gamma_{RLG}, {}_j\Rightarrow) \subseteq \mathscr{L}(\Gamma_{LG}, {}_j\Rightarrow) \subseteq \bigcup_{k\geq 1} \mathscr{L}(\Gamma_{CFG}, {}_j\Rightarrow_k)$ follows from the definitions, it suffices to proof that $\bigcup_{k\geq 1} \mathscr{L}(\Gamma_{CFG}, {}_j\Rightarrow_k) \subseteq \mathscr{L}(\Gamma_{RLG}, {}_j\Rightarrow)$.

Construction. Let V and T be an alphabet and an alphabet of terminals, respectively. Set $N = V - T$. Let $\eta\colon V \to N \cup \{\varepsilon\}$ be a homomorphism such that $\eta(X) = X$ if $X \in N$; otherwise $\eta(X) = \varepsilon$. Let $\tau\colon V \to T \cup \{\varepsilon\}$ be a homomorphism such that $\tau(X) = X$ if $X \in T$; otherwise $\eta(X) = \varepsilon$. As usual, extend η and τ to strings of symbols.

For every CFG $G = (V_G, T, P_G, S)$ and index $k \geq 1$, we construct a RLG $H = (V_H, T, P_H, \langle S\rangle)$ such that $L(G, {}_j\Rightarrow_k) = L(H, {}_j\Rightarrow)$. Set

$$V_H = \{\langle x\rangle \mid x \in \bigcup_{i=1}^{k} (V_G - T)^i\} \cup T$$

and set

$$P_H = \{\langle \alpha A\beta\rangle \to \tau(x)\langle\gamma\rangle \mid A \to x \in P_G, \alpha, \beta \in N^*, \gamma = \alpha\beta\eta(x), 1 \leq |\gamma| \leq k\}$$
$$\cup \{\langle A\rangle \to x \mid A \to x \in P_G, x \in T^*\}$$

Basic Idea. CFG G working with index k means that every sentential form contains at most k nonterminal symbols. In jumping derivation mode, the position of nonterminal symbol does not matter for context-free rewriting. Together with the finiteness of N, we can store the list of nonterminals using just one nonterminal from constructed $V_H - T$ in the simulating RLG.

For every jumping derivation step $\gamma A\delta\ {}_j\Rightarrow_k \gamma' x\delta'$ by $A \to x$ in G, there is a simulating jumping derivation step $\tau(\bar{\gamma})\langle\eta(\gamma A\delta)\rangle\tau(\bar{\delta})\ {}_j\Rightarrow \tau(\bar{\gamma'})\tau(x)\langle\eta(\gamma\delta x)\rangle\tau(\bar{\delta'})$ in H where $\gamma\delta = \gamma'\delta' = \bar{\gamma}\bar{\delta} = \bar{\gamma'}\bar{\delta'}$. The last simulating step of jumping application of $A \to w$ with $w \in T^*$ replaces the only nonterminal of the form $\langle A\rangle$ by w that can be placed anywhere in the string. □

Consider the finite index restriction in the family $\bigcup_{k\geq 1} \mathscr{L}(\Gamma_{CFG}, {}_j\Rightarrow_k)$ in Theorem 15. Dropping this restriction gives rise to the next question. Indeed, from a broader perspective, an investigation of finite-index-based restrictions placed upon various jumping grammars and their effect on the resulting generative power represents a challenging open problem area as illustrated by Example 9.

Open Problem 16. *Is* $\bigcup_{k\geq 1} \mathscr{L}(\Gamma_{CFG}, {}_j\Rightarrow_k) \subseteq \mathscr{L}(\Gamma_{CFG}, {}_j\Rightarrow)$ *proper?*

Theorem 17. $\mathscr{L}(\Gamma_{CFG^{-\varepsilon}}, {}_j\Rightarrow) = \mathscr{L}(\Gamma_{CFG}, {}_j\Rightarrow)$.

Proof. It is easy to prove this theorem by analogy with the theorem for CFGs in sequential derivation mode (see Theorem 5.1.3.2.4 on page 328 in [8]). □

Lemma 18. $\mathbf{RE} \subseteq \mathscr{L}(\Gamma_{GG}, {}_j\Rightarrow)$.

Proof.
Construction. For every GG $G = (V_G, T, P_G, S_G)$, we construct another GG $H = (V_H = V_G \cup \{S_H, \$, \#, \lfloor, \rfloor\}, T, P_H, S_H)$ such that $L(G, {}_s\Rightarrow) = L(H, {}_j\Rightarrow)$. $S_H, \$, \#, \lfloor$, and $\rfloor$ are new nonterminal symbols in H. Set

$$P_H = \{S_H \to \#S_G, \# \to \lfloor\$, \lfloor\ \rfloor \to \#, \# \to \varepsilon\} \cup \{\$\alpha \to \rfloor\beta \mid \alpha \to \beta \in P_G\}$$

Basic Idea. Nonterminal # has at most one occurrence in the sentential form. # is generated by the initial production $S_H \to \#S_G$. This symbol participates in the beginning and end of every simulation of the application of a production from P_G. Each simulation consists of several jumping derivation steps:

(i) # is expanded to a string of two nonterminals—marker of a position ($\lfloor$), where the production is applied in the sentential form, and auxiliary symbol (\$) presented as a left context symbol in the left-hand side of every simulated production from P_G.
(ii) For each $x \to y$ from P_G, $\$x \to \rfloor y$ is applied in H. To be able to finish the simulation properly, the right-hand side ($\rfloor y$) of applied production has to be placed right next to the marker symbol $\lfloor$; otherwise, we cannot generate a sentence.
(iii) The end of the simulation (production $\lfloor \rfloor \to \#$) checks that the jumping derivation was applied like in the sequential way.
(iv) In the end, # is removed to finish the generation of a string of terminal symbols.

Define the homomorphism $h\colon V_H \to V_G$ as $h(X) = X$ for all $X \in V_G$, $h(S_H) = S_G$, and $h(Y) = \varepsilon$ for all $Y \in \{\$, \#, \lfloor, \rfloor\}$.

Claim 19. *Let y be a sentential form of H; that is, $S_H \Rightarrow_j^* y$. For every $X \in \{\#, \$, \lfloor, \rfloor, S_H\}$, occur$(\{X\}, y) \leq 1$.*

Proof. The claim follows from the productions in P_H constructed in the proof of the previous lemma. In addition, note that occur$(\{\#, \$, \lfloor, \lfloor, S_H\}, y) \leq 2$ and symbol # occurs in y exclusively with respect to \$, $\lfloor$, $\rfloor$, and S_H. □

Claim 20. *If $S_G \Rightarrow_s^k w$ in G, where $w \in T^*$ and $k \geq 0$, then $S_H \Rightarrow_j^* w$ in H.*

Proof. First, we prove by induction on $k \geq 0$ that for every $S_G \Rightarrow_s^k x$ in G with $x \in V_G^*$, there is $S_H \Rightarrow_j^* x'$ in H such that $h(x') = x$.
Basis. For $S_G \Rightarrow_s^0 S_G$ in G, there is $S_H \Rightarrow_j \#S_G$ in H.
Induction Hypothesis. For some $k \geq 0$, $S_G \Rightarrow_s^k x$ in G implies that $S_H \Rightarrow_j^* x'$ in H such that $h(x') = x$.
Induction Step. Assume that $S_G \Rightarrow_s^k y \Rightarrow_s x$ in G. By induction hypothesis, $S_H \Rightarrow_j^* y'$ in H with $h(y') = y$.

The derivation step $y \Rightarrow_s x$ in G is simulated by an application of three jumping productions from P_H in H to get $y' \Rightarrow_j^3 x'$ with $h(x') = x$ as follows.

$$\begin{aligned} y' = u'\#v' &\Rightarrow_j u''\lfloor\$\alpha v'' && [\# \to \lfloor\$] \\ &\Rightarrow_j u''\lfloor\rfloor\beta v'' && [\lfloor\rfloor \to \#] \\ &\Rightarrow_j u'''\#v''' && [\# \to \varepsilon] \quad = x' \end{aligned}$$

where $u'v' = u''\alpha v''$ and $u''\beta v'' = u'''v'''$.

In case $x \in T^*$, there is one additional jumping derivation step during the simulation that erases the only occurrence of #-symbol (see Claim 19) by production $\# \to \varepsilon$.

Note that $h(x)$ for $x \in T^*$ is the identity. Therefore, in case $x \in T^*$ the induction proves the claim. □

Claim 21. *If $S_H \,{}_j\!\Rightarrow^k w$ in H, where $w \in T^*$, then $S_G \,{}_s\!\Rightarrow^* w$ in G.*

Proof. To prove this claim, first, we prove by induction on $k \geq 0$ that for every $S_H \,{}_j\!\Rightarrow^k x$ in H with $x \in V_H^*$ such that there exists a jumping derivation $x \,{}_j\!\Rightarrow^* w$, where $w \in T^*$, then $S_G \,{}_s\!\Rightarrow^* x'$ in G such that $h(x) = x'$.
Basis. For $k = 0$, we have $S_H \,{}_j\!\Rightarrow^0 S_H \,{}_j\!\Rightarrow^* w$ in H, then there is $S_G \,{}_s\!\Rightarrow^0 S_G$ in G such that $h(S_H) = S_G$. Further, for $k = 1$, we have $S_H \,{}_j\!\Rightarrow^1 \#S_G \,{}_j\!\Rightarrow^* w$ in H, then again there is $S_G \,{}_s\!\Rightarrow^0 S_G$ in G such that $h(\#S_G) = S_G$, so the basis holds.
Induction Hypothesis. For some $k \geq 1$, $S_H \,{}_j\!\Rightarrow^k x \,{}_j\!\Rightarrow^* w$ in H implies that $S_G \,{}_s\!\Rightarrow^* x'$ in G such that $h(x) = x'$.
Induction Step. Let $u, v \in V_G^*$ and $\bar{u}, \bar{v} \in V_H^*$. Assume that $S_H \,{}_j\!\Rightarrow^k y \,{}_j\!\Rightarrow x \,{}_j\!\Rightarrow^* w$ in H with $w \in T^*$. By induction hypothesis, $S_G \,{}_s\!\Rightarrow^* y'$ in G such that $h(y) = y'$. Let us examine the following possibilities of $y \,{}_j\!\Rightarrow x$ in H:

(i) $y = u\#v \,{}_j\!\Rightarrow \bar{u}\lfloor\$\bar{v} = x$ in H such that $uv = \bar{u}\bar{v}$: Simply, $y' = uv \,{}_s\!\Rightarrow^0 uv$ in G and by Claim 19 $h(\bar{u}\lfloor\$\bar{v}) = h(\bar{u}\bar{v}) = h(uv) = uv$.
(ii) $u\lfloor\$\alpha v \,{}_j\!\Rightarrow \bar{u}\rfloor\beta\bar{v}$ in H by production $\$\alpha \rightarrow \rfloor\beta$ such that $uv = \bar{u}\bar{v}$: In fact, to be able to rewrite $\lfloor$, the symbol $\lfloor$ needs $\rfloor$ as its right neighbor, so $u = \bar{u}$ and $v = \bar{v}$ in this jumping derivation step; otherwise the jumping derivation is blocked and string of terminals cannot be generated. According to production $\alpha \rightarrow \beta$, $u\alpha v \,{}_s\!\Rightarrow u\beta v$ in G and $h(\bar{u}\rfloor\beta\bar{v}) = u\beta v$.
(iii) $u\lfloor\rfloor v \,{}_j\!\Rightarrow \bar{u}\#\bar{v}$ in H such that $uv = \bar{u}\bar{v}$: In G, $uv \,{}_s\!\Rightarrow^0 uv$ and $h(\bar{u}\#\bar{v}) = h(\bar{u}\bar{v}) = h(uv) = uv$.
(iv) $u\#v \,{}_j\!\Rightarrow uv$ in H by $\# \rightarrow \varepsilon$: Trivially, $uv \,{}_s\!\Rightarrow^0 uv$ in G and $h(uv) = uv$.

If $x \in T^*$, then the induction proves the claim. □

Theorem 22. $\mathscr{L}(\Gamma_{GG}, \,{}_j\!\Rightarrow) = \mathbf{RE}$.

Proof. Trivially, by Turing-Church thesis, $\mathscr{L}(\Gamma_{GG}, \,{}_j\!\Rightarrow) \subseteq \mathbf{RE}$. The opposite inclusion holds by Lemma 18 that is proved in details by Claim 20 and Claim 21. □

3.3. *Properties of Jumping Derivations*

We demonstrate that the order of nonterminals in a sentential form of jumping CFGs is irrelevant. Then, in this section, we study the semilinearity of language families generated by various jumping grammars.

As a generalization of the proof of Theorem 15, we give the following lemma demonstrating that the order in which nonterminals occur in sentential forms is irrelevant in jumping derivation mode based on context-free productions in terms of generative power.

Lemma 23. *Let η and τ be the homomorphisms from the proof of Theorem 15. For every $G \in \Gamma_X$ with $X \in \{$RG, RLG, LG, CFG$\}$ and $G = (V, T, P, S)$ with $N = V - T$, if $S \,{}_j\!\Rightarrow^* \gamma \,{}_j\!\Rightarrow^k w$ in G, $k \geq 0$, $\gamma \in V^*$, $w \in T^*$, then for every $\delta \in V^*$ such that $\tau(\gamma) = \tau(\delta)$ and $\eta(\delta) \in$ perm$(\eta(\gamma))$, there is $\delta \,{}_j\!\Rightarrow^* w$ in G.*

Proof. We prove this lemma by induction on $k \geq 0$.

Basis. Let $k = 0$. That is, $S \ {}_j\Rightarrow^* \gamma \ {}_j\Rightarrow^0 w$ in G, so $\gamma = w$. By $\tau(\delta) = \tau(\gamma)$, we have $\gamma = w = \delta$, so $\delta \ {}_j\Rightarrow^0 w$ in G.

Induction Hypothesis. Assume that the lemma holds for some $k \geq 0$.

Induction Step. Assume that $S \ {}_j\Rightarrow^* \gamma \ {}_j\Rightarrow \gamma' \ [A \to x] \ {}_j\Rightarrow^k w$ in G with $k \geq 0$. Observe that $\tau(\delta) = \tau(\gamma)$ and $\eta(\delta) \in \text{perm}(\eta(\gamma))$. By the above-mentioned assumption, $|\eta(\gamma)| \geq 1$—that is $|\eta(\delta)| \geq 1$. Thus, the jumping derivation $\delta \ {}_j\Rightarrow^* w$ in G can be written as $\delta \ {}_j\Rightarrow \delta' \ [A \to x] \ {}_j\Rightarrow^* w$. Since all the productions in G are context-free, the position of A in δ and its context is irrelevant, and the occurrence of A in δ is guaranteed by the lemma precondition. During the application of $A \to x$, (1) an occurrence of A is found in δ, (2) removed, and (3) the right-hand side of the production, x, is inserted anywhere in δ instead of A without preserving the position of the rewritten A. Assume x is inserted into δ' so that $\tau(\delta') = \tau(\gamma')$. We also preserve that $\eta(\delta') \in \text{perm}(\eta(\gamma'))$; therefore, the lemma holds. □

Notice that even if there is no derivation $S \ {}_j\Rightarrow^* \delta$ in G, the lemma holds.

Note that based on the proof of Lemma 23, we can turn any jumping version of a CFG to an equivalent jumping CFG satisfying a modified Greibach normal form, in which each production is of the form $A \to \alpha\beta$ where $\alpha \in T^*, \beta \in N^*$. Observe that $\alpha \notin T$. Consider, for instance, a context-free production p with $\alpha = a_1 \cdots a_n$. By an application of p during a derivation of a string of terminals w, we arrange that a_1 appears somewhere in front of a_n in w. In other words, from Theorem 13 and Corollary 14 in [9] together with Theorem 14 above, it follows for any language L, $L \in \mathscr{L}(\Gamma_{RG}, \ {}_j\Rightarrow)$ implies $L = \text{perm}(L)$, which means that the order of all terminals in $w \in L$ is utterly irrelevant.

Corollary 24. *For every $G \in \Gamma_X$ with $X \in \{$RG, RLG, LG, CFG$\}$, $S \ {}_j\Rightarrow^* \gamma \ {}_j\Rightarrow^* w$ in G implies an existence of a derivation of the following form*

$$S \ {}_j\Rightarrow^* \alpha\beta \ {}_j\Rightarrow^* w \text{ in } G$$

where $\alpha = \tau(\gamma)$, $\beta \in$ perm$(\eta(\gamma))$, S is the start nonterminal, and w is a string of terminals.

Definition 25. *([4]) Let $w \in V^*$ with $V = \{a_1, \ldots, a_n\}$. We define* Parikh vector *of w by $\psi_V(w) = (occur(a_1, w), occur(a_2, w), \ldots, occur(a_n, w))$. A set of vectors is called* semilinear *if it can be represented as a union of a finite number of sets of the form $\{v_0 + \sum_{i=1}^{m} \alpha_i v_i \mid \alpha_i \in \mathbb{N}, 1 \leq i \leq m\}$ where v_i for $0 \leq i \leq m$ is an n-dimensional vector. A language $L \subseteq V^*$ is called* semilinear *if the set $\psi_V(L) = \{\psi_V(w) \mid w \in L\}$ is a semilinear set. A language family is* semilinear *if all its languages are semilinear.*

Lemma 26. *For $X \in \{$RG, RLG, LG, CFG$\}$, $\mathscr{L}(\Gamma_X, \ {}_j\Rightarrow)$ is semilinear.*

Proof. By Parikh's Theorem (see Theorem 6.9.2 on page 228 in [5]), for each context-free language $L \subseteq V^*$, $\psi_V(L)$ is semilinear. Let G be a CFG such that $L(G, \ {}_s\Rightarrow) = L$. From the definition of ${}_j\Rightarrow$ and CFG it follows that $\psi(L(G, \ {}_s\Rightarrow)) = \psi(L(G, \ {}_j\Rightarrow))$ therefore $\psi(L(G, \ {}_j\Rightarrow))$ is semilinear as well. □

Recall that the family of context-sensitive languages is not semilinear (for instance, Example 2.3.1 and Theorem 2.3.1 in [3] implies that $\{a^{2^n} \mid n \geq 0\} \in \mathbf{CS}$, but is not semilinear language). By no means, this result rules out that $\mathscr{L}(\Gamma_{CSG}, {}_j\Rightarrow)$ or $\mathscr{L}(\Gamma_{MONG}, {}_j\Rightarrow)$ are semilinear. There is, however, another kind of results concerning multiset grammars (see [6]) saying that a context-sensitive multiset grammar generates a non-semilinear language. The multiset grammars work with Parikh vector of a sentential form so the order of symbols in the sentential form is irrelevant. Then, all permutations of terminal strings generated by the grammar belong to the generated language.

Instead of the full definition of multiset grammars (see [6]), based on notions from the theory of macrosets, we introduce *multiset derivation mode* concerning the classical string formal language theory.

Definition 27. *Let $u, v \in V^*$ and $G = (V, T, P, S)$ be a grammar. $u\ {}_m\Rightarrow v\ [x \to y]$ in G iff there exist $x \to y \in P$ and $t, t', z, z' \in V^*$ such that $txt' \in perm(u)$ and $zyz' \in perm(v)$.*

Recall that $\mathscr{L}(\Gamma_{MONG}, {}_m\Rightarrow)$ is not semilinear (see [6]). As every context-sensitive multiset grammar can be transformed into CSG that generates the same language under jumping derivation mode, we can prove the following theorem.

Theorem 28. *Both $\mathscr{L}(\Gamma_{CSG}, {}_j\Rightarrow)$ and $\mathscr{L}(\Gamma_{MONG}, {}_j\Rightarrow)$ are not semilinear.*

Proof. In [6], there is a result implying that $\mathscr{L}(\Gamma_{MONG}, {}_m\Rightarrow)$ contains non-semilinear languages, so to proof non-semilinearity of $\mathscr{L}(\Gamma_{CSG}, {}_j\Rightarrow)$ and $\mathscr{L}(\Gamma_{MONG}, {}_j\Rightarrow)$, it is enough to prove that $\mathscr{L}(\Gamma_{MONG}, {}_m\Rightarrow) \subseteq \mathscr{L}(\Gamma_{CSG}, {}_j\Rightarrow)$ because $\mathscr{L}(\Gamma_{CSG}, {}_j\Rightarrow) \subseteq \mathscr{L}(\Gamma_{MONG}, {}_j\Rightarrow)$ by Definition 1.

Construction. For every MONG $G = (V_G, T, P_G, S)$, we construct a CSG $H = (V_H, T, P_H, S)$ such that $L(G, {}_m\Rightarrow) = L(H, {}_j\Rightarrow)$. Let h be a homomorphism $h\colon V_G \to V_H$ as $h(X) = X$ for all $X \in V_G - T$ and $h(a) = \langle a \rangle$ for all $a \in T$. First, set $V_H = V_G \cup \{\langle a \rangle \mid a \in T\}$ and $P_t = \{\langle a \rangle \to a \mid a \in T\}$. We initialize P_{cf} to all context-free productions from P_G such that all terminals are replaced with special nonterminals; that is, $P_{cf} = \{A \to h(x) \mid A \to x \in P_G,\ A \in V_G - T \text{ and } x \in V_G^*\}$. Set $P_{cs} = \emptyset$. For every production $p\colon\ X_1X_2\cdots X_n \to Y_1Y_2\cdots Y_m \in P_G$ with $2 \leq n \leq m$, where $X_i, Y_j \in V_G$, $1 \leq i \leq n$ and $1 \leq j \leq m$, we add $2n$ new productions into P_{cs}:

$$
\begin{array}{rrcl}
p_1\colon & h(X_1X_2\cdots X_n) & \to & h(X_{1,p}X_2\cdots X_n) \\
p_2\colon & h(X_{1,p}X_2\cdots X_n) & \to & h(X_{1,p}X_{2,p}\cdots X_n) \\
 & & \vdots & \\
p_n\colon & h(X_{1,p}X_{2,p}\cdots X_n) & \to & h(X_{1,p}X_{2,p}\cdots X_{n,p}) \\
p_{n+1}\colon & h(X_{1,p}X_{2,p}\cdots X_{n,p}) & \to & h(Y_1X_{2,p}\cdots X_{n,p}) \\
p_{n+2}\colon & h(Y_1X_{2,p}\cdots X_{n,p}) & \to & h(Y_1Y_2\cdots X_{n,p}) \\
 & & \vdots & \\
p_{2n}\colon & h(Y_1Y_2\cdots X_{n,p}) & \to & h(Y_1Y_2\cdots Y_nY_{n+1}\cdots Y_m)
\end{array}
$$

and add new nonterminals $X_{i,p}$, $1 \leq i \leq n$ into V_H. According to the final V_H, set $P_c = \{A \to A \mid A \in V_H - T\}$. Now, set $P_H = P_{cf} \cup P_t \cup P_c \cup P_{cs}$.

Basic Idea. A CSG can rewrite only one nonterminal at once in the given unchangeable left and right context. In addition, the adjacency of symbols to be rewritten by a production is not required in multiset derivation mode; therefore, first, by using homomorphism h, we introduce new nonterminals of form $\langle a \rangle$, $a \in T$, and, thereby, replace the terminals until the final jumping derivation phase of the simulation of a multiset derivation. Then, in any simulating sentential form, by using productions from P_c, we can change the order of all nonterminals so the adjacency is not a restriction in H either, and we only require the occurrence of the symbols from $\text{lhs}(p)$ during the simulation of application of $p \in P_G$.

An application of a monotonous context-sensitive production $p\colon X_1X_2 \cdots X_n \to Y_1Y_2 \cdots Y_m \in P_G$, $2 \leq n \leq m$ in $u \ {}_m\Rightarrow v\ [p]$ in G is simulated in several steps in H as follows:

$$
\begin{array}{lcll}
u = u_0 & {}_j\Rightarrow^* & \alpha_0 X_1 X_2 \cdots X_n \beta_0\ [\rho_0] & {}_j\Rightarrow \quad u_1\ [p_1] \\
 & {}_j\Rightarrow^* & \alpha_1 X_{1,p} X_2 \cdots X_n \beta_1\ [\rho_1] & {}_j\Rightarrow \quad u_2\ [p_2] \\
 & {}_j\Rightarrow^* & \alpha_2 X_{1,p} X_{2,p} \cdots X_n \beta_2\ [\rho_2] & {}_j\Rightarrow \quad u_3\ [p_3] \\
 & \vdots & & \\
 & {}_j\Rightarrow^* & \alpha_{n-1} X_{1,p} X_{2,p} \cdots X_n \beta_{n-1}\ [\rho_{n-1}] & {}_j\Rightarrow \quad u_n\ [p_n] \\
 & {}_j\Rightarrow^* & \alpha_n X_{1,p} X_{2,p} \cdots X_{n,p} \beta_n\ [\rho_n] & {}_j\Rightarrow \quad u_{n+1}\ [p_{n+1}] \\
 & \vdots & & \\
 & {}_j\Rightarrow^* & \alpha_{2n-1} Y_1 Y_2 \cdots X_{n,p} \beta_{2n-1}\ [\rho_{2n-1}] & {}_j\Rightarrow \quad u_{2n}\ [p_{2n}] \\
 & = & \alpha_{2n} Y_1 Y_2 \cdots Y_m \beta_{2n} & {}_j\Rightarrow^* \quad v\ [\rho_{2n}]
\end{array}
$$

where $\rho_i \in P_c^*$, $0 \leq i \leq 2n$.

In $u_0 \ {}_j\Rightarrow^* u_n$, we verify in G that u contains all symbols from $\text{lhs}(p)$ and we mark them by subscript p. Then, by $u_n \ {}_j\Rightarrow^* v$ we simulate the rewritting by p.

In the final phase of the simulation, the nonterminals $\langle a \rangle$ are replaced by corresponding terminals using productions from P_t.

The technical proof of $L(G, {}_m\Rightarrow) = L(H, {}_j\Rightarrow)$ is left to the reader. □

4. Conclusion

In this final section, we propose several future investigation areas concerning jumping grammars. Some of them relate to specific open questions pointed out earlier in the paper; the present section, however, formulates them more generally and broadly.

I. *Other Types of Grammars*. The present paper has concentrated its attention to the language families resulting from classical grammars, such as the grammars classified by Chomsky. Apart from them, however, the formal language theory has introduced many other types of grammars, ranging from regulated grammars through parallel grammars up to grammar systems. Reconsider the present study in their terms.

II. *Left and Right Jumping Mode*. Considering the left and right jumps of JFAs and GJFAs in [9], we are inspired to an introduction and discussion of left and right jumping

derivation modes (see (ii) and (iii) in Definition 1) in terms of classical types of grammars.

III. *Closure Properties*. Several results and some open problems concerning closure properties of **JFA** and **GJFA** are stated in [9]. Considering these results as well as Theorems 13 and 14 above, study closure properties of language families generated in a jumping way. Specifically, investigate these properties in terms of CFGs, CSGs, and $MONG$s.

IV. *Alternative Definition of Jumping Mode with Context*. Assume context-sensitive productions (CSG) of the following form

$$\alpha A\beta \to \alpha\gamma\beta, \text{ where } A \in N, \alpha, \beta, \gamma \in V^*, \gamma \neq \varepsilon$$

There are three interesting ways of defining a jumping derivation step:

(a) Using the previous definition (see Definition 1) of jumping derivation; that is, find $\alpha A\beta$ in the current sentential form $u\alpha A\beta v$, remove $\alpha A\beta$, and place $\alpha\gamma\beta$ anywhere in uv. For instance,

$$aAbc \;{}_{j}\Rightarrow caxb \; [aAb \to axb]$$

(b) Do not move the context of the rewritten nonterminal; that is, find A with left context α and right context β, remove this A from the current sentential form, and place γ in the new sentential form, such that string γ will be again in the context of both α and β (but it can be different occurrence of α and β). For instance,

$$aAbab \;{}_{j'}\Rightarrow abaxb \; [aAb \to axb]$$

(c) Similarly to (b), in the third variant we do not move the context of the rewritten nonterminal either and, in addition, γ has to be placed between the same occurrence of α and β. As a consequence, context-sensitive productions are applied sequentially even in this jumping derivation mode. For instance,

$$aAbab \;{}_{j''}\Rightarrow axbab \; [aAb \to axb]$$

Notice that this derivation mode influences only the application of context-free productions (i.e. $\alpha = \beta = \varepsilon$).

Example 29. *Example 4 shows a CSG that generates* $\{a^n b^n \mid n \geq 1\}$ *when the alternative jumping derivation mode* ${}_{j'}\Rightarrow$ *for CSGs is used. In context of Lemma 5, the alternative jumping derivation mode (b) can increase the generative power of jumping CSGs (a). In fact, it is an open question whether* $\mathscr{L}(\Gamma_{CSG}, {}_{j'}\Rightarrow) \subseteq \mathscr{L}(\Gamma_{MONG}, {}_{j'}\Rightarrow)$.

V. *Relationship with Formal Macroset Theory*. Recently, formal language theory has introduced various rewriting devices that generate different objects than classical formal languages. Specifically, in this way, Formal Macroset Theory has investigated the generation of macrosets—that is, sets of multisets over alphabets. Notice that some of its results resemble results achieved in the present study (c.f., for instance, Theorem 1 in

[6] and Theorem 15 and Open Problem 16 above). Explain this resemblance mathematically.

Acknowledgments

This work was supported by the European Regional Development Fund in the IT4Innovations Centre of Excellence project (CZ.1.05/1.1.00/02.0070), the TAČR grant TE01010415, and the BUT FIT FIT-S-11-2 grant.

References

[1] R. Baeza-Yates and B. Ribeiro-Neto, *Modern Information Retrieval: The Concepts and Technology behind Search*, (Addison-Wesley, Boston, 2nd edition, 2011).
[2] S. Buettcher, C. L. A. Clarke, and G. V. Cormack, *Information Retrieval: Implementing and Evaluating Search Engines*, (The MIT Press, Cambridge, 2010).
[3] J. Dassow and Gh. Păun, *Regulated Rewriting in Formal Language Theory* (Springer, Berlin, 1989).
[4] S. Ginsburg, *The Mathematical Theory of Context-free Languages* (McGraw Hill, New York, 1966).
[5] M. A. Harrison, *Introduction to Formal Language Theory* (Addison-Wesley, Boston, 1978).
[6] M. Kudlek, C. Martín-Vide, and Gh. Păun, Toward FMT (Formal Macroset Theory), In: *Preproceedings of the Workshop on Multiset Processing* (Curtea de Arges, August 21-25, 2000), pages 149-158.
[7] C. D. Manning, P. Raghavan, and H. Schütze, *Introduction to Information Retrieval*, (Cambridge University Press, New York, 2008).
[8] A. Meduna, *Automata and Languages* (Springer, London, 2000).
[9] A. Meduna and P. Zemek, Jumping Automata. *Int. J. Found. Comput. Sci.* **23**(2012) 1555–1578. doi:10.1142/S0129054112500244.
[10] A. Salomaa, *Formal Languages* (Academic Press, 1973).
[11] D. Wood, *Theory of Computation: A Primer* (Addison-Wesley, Boston, 1987).