ON DOUBLE-JUMPING FINITE AUTOMATA

Radim Kocman Zbyněk Křivka
Alexander Meduna

Centre of Excellence IT4Innovations
Faculty of Information Technology
Brno University of Technology
Božetěchova 2, Brno Czech Republic
{ikocman,krivka,meduna}@fit.vutbr.cz

Abstract
The present paper modifies and studies jumping finite automata so they always perform two simultaneous jumps according to the same rule. For either of the two simultaneous jumps, it considers three natural directions—(1) to the left, (2) to the right, and (3) in either direction. According to this jumping-direction three-part classification, the paper investigates the mutual relation between the language families resulting from jumping finite automata performing the jumps in these ways and the families of regular, linear, context-free, and context-sensitive languages. It demonstrates that most of these language families are pairwise incomparable—that is, they are not subfamilies of each other and, simultaneously, they are not disjoint.

1. Introduction

At present, jumping versions of rewriting systems, such as grammars and automata, represent a vivid investigation area in language theory (see [1, 2, 3, 4, 6, 8, 9]). In essence, they work just like classical rewriting systems except that they work on strings discontinuously. That is, they apply a production so they erase an occurrence of its left-hand side in the rewritten string while placing the right-hand side anywhere in the string. As a result, the position of the insertion may occur far away from the position of the erasure. The present paper continues with this investigation in terms of jumping finite automata.

To give an insight into this study, let us first recall the well-known notion of a classical finite automaton, M, which consists of an input tape, a read head, and a finite state control. The input tape is divided into squares. Each square contains one symbol of an input string. The symbol under the read head, a, is the current input symbol. The finite control is represented by a finite set of states together with a control relation, which is usually specified as a set of computational rules. M computes by making a sequence of moves. Each move is made according to a computational rule that describes how the current state is changed and whether

the current input symbol is read. If the symbol is read, the read head is shifted precisely one square to the right. M has one state defined as the start state and some states designated as final states. If M can read w by making a sequence of moves from the start state to a final state, M accepts w; otherwise, M rejects w.

In essence, a jumping finite automaton works just like a classical finite automaton except it does not read the input string in a symbol-by-symbol left-to-right way. That is, after reading a symbol, M can jump over a portion of the tape in either direction and continue making moves from there. Once an occurrence of a symbol is read on the tape, it cannot be re-read again later during the computation of M. Otherwise, it coincides with the standard notion of a finite automaton.

Consider the notion of a jumping finite automaton M sketched above. The present paper modifies the way M works so it simultaneously performs two jumps according to the same rule. For either of the two jumps, it always considers three natural directions—(1) to the left, (2) to the right, and (3) in either direction. In correspondence to this jumping-direction three-part classification, the paper investigates the mutual relation between the language families resulting from jumping finite automata working in these ways and the families of regular, linear, context-free, and context-sensitive languages. In essence, it demonstrates that most of these language families are pairwise incomparable—that is, they are not subfamilies of each other and, simultaneously, they are not disjoint either. Consequently, from a computationally broader viewpoint, it actually demonstrates that the jumping directions fulfill an essential role in the computation formalized by jumping finite automata—a general result, which might be of some interest and importance to the investigation as well as the application of jumping finite automata in the future.

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 modes of general jumping finite automata. Section 3 presents fundamental results achieved in this paper. Section 4 closes all the study by pointing out some open problem areas.

2. Preliminaries and Definitions

This paper assumes that the reader is familiar with the theory of automata and formal languages (see [5, 10]). For a set Q, $card(Q)$ denotes the cardinality of Q. For an alphabet (finite nonempty set) V, V^* represents the free monoid generated by V under the operation of concatenation. The unit of V^* is denoted by ε. Members of V^* are called *strings*. For $x \in V^*$, $|x|$ denotes the length of x, and $alph(x)$ denotes the set of all symbols occurring in x; for instance, $alph(0010) = \{0, 1\}$. Let $x = a_1 a_2 \dots a_n$, where $a_i \in V$ for all $i = 1, \dots, n$, for some $n \geq 0$ ($x = \varepsilon$ if and only if $n = 0$). Let X and Y be sets; we call X and Y to be *incomparable* if $X \not\subseteq Y$, $Y \not\subseteq X$, and $X \cap Y \neq \emptyset$.

A *linear grammar* is a quadruple $G = (N, T, P, S)$, where N and T are alphabets such that $N \cap T = \emptyset$, $S \in N$, and P is a finite set of productions of the form $A \to x$, where $A \in N$

and $x \in T^*(N \cup \{\varepsilon\})T^*$. If $A \to x \in P$ and $u, v \in T^*$, then $uAv \Rightarrow uxv$ $[A \to x]$, or simply $uAv \Rightarrow uxv$. In the standard manner, extend $\Rightarrow$ to $\Rightarrow^n$, where $n \geq 0$; then, based on $\Rightarrow^n$, define $\Rightarrow^+$ and $\Rightarrow^*$. The language of G, $L(G)$, is defined as $L(G) = \{w \in T^* \mid S \Rightarrow^* w\}$. A language, L, is linear if and only if $L = L(G)$, where G is a linear grammar.

A *right-linear grammar* is a linear grammar $G = (N, T, P, S)$ such that every rule in P is of the form $A \to x$ and satisfies $x \in T^*N \cup T^*$. A language, L, is right-linear (or regular) if and only if $L = L(G)$, where G is a right-linear grammar.

A *general jumping finite automaton* (see [6]), a *GJFA* for short, is a quintuple $M = (Q, \Sigma, R, s, F)$, where Q is a finite set of states, Σ is an input alphabet, $Q \cap \Sigma = \emptyset$, $R \subseteq Q \times \Sigma^* \times Q$ is finite, $s \in Q$ is the start state, and F is a set of final states. Members of R are referred to as rules of M. For brevity, we sometimes denote a rule (p, y, q) with a unique label h as $h : (p, y, q)$, and instead of $h : (p, y, q) \in R$, we simply write $h \in R$. 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 $(p, y, q) \in R$; then, M makes a *jump* from $xpyz$ to $x'qz'$, symbolically written as $xpyz \curvearrowright x'qz'$.

We define a new mode for general jumping finite automata that performs two single jumps simultaneously. In this mode, both single jumps follow the same rule, however, they are performed on two different positions on the tape and thus handle different parts of the input string. Moreover, these two jumps cannot ever cross each other—their initial mutual order is preserved during the whole process. As a result, when needed, we can specifically denote them as the first jump and the second jump. Furthermore, we define new versions of single jumps in order to get a more consistent behavior. These modified single jumps always read strings from the tape in the same direction as the direction of their jumping.

Let $M = (Q, \Sigma, R, s, F)$ be a GJFA. Let X denote the set of all configurations of M. Let $w, x, y, z \in \Sigma^*$ and $h : (p, y, q) \in R$; then,

$$wpyxz \ {}_{\blacktriangleright}\curvearrowright \ wxqz \ [h]$$

in M. When the specification of the rule h is immaterial, we can omit $[h]$. Let $w, x, y, z \in \Sigma^*$ and $h : (p, y, q) \in R$; then,

$$wxypz \ {}_{\blacktriangleleft}\curvearrowright \ wqxz \ [h]$$

in M. The binary *unrestricted jumping relation*, symbolically denoted by ${}_{\blacklozenge}\curvearrowright$, over X, is defined as follows. Let $\zeta, \vartheta \in X$, and $h \in R$; then, M makes an *unrestricted jump* from ζ to ϑ according to h, symbolically written as

$$\zeta \ {}_{\blacklozenge}\curvearrowright \ \vartheta \ [h]$$

if either $\zeta \ {}_{\blacktriangleright}\curvearrowright \vartheta \ [h]$ or $\zeta \ {}_{\blacktriangleleft}\curvearrowright \vartheta \ [h]$.

A 2-*configuration* of M is any string in XX. Let X^2 denote the set of all 2-configurations of M. The binary *unrestricted* 2-*jumping relation*, symbolically denoted by ${}_{\blacklozenge\blacklozenge}\curvearrowright$, over X^2, is defined as follows. Let $\zeta_1\zeta_2, \vartheta_1\vartheta_2 \in X^2$, where $\zeta_1, \zeta_2, \vartheta_1, \vartheta_2 \in X$, and $h \in R$; then, M makes an *unrestricted* 2-*jump* from $\zeta_1\zeta_2$ to $\vartheta_1\vartheta_2$ according to h, symbolically written as

$$\zeta_1\zeta_2 \ {}_{\blacklozenge\blacklozenge}\curvearrowright \ \vartheta_1\vartheta_2 \ [h]$$

if and only if $\zeta_1 \ {}_{\blacklozenge}\!\curvearrowright \vartheta_1$ $[h]$ and $\zeta_2 \ {}_{\blacklozenge}\!\curvearrowright \vartheta_2$ $[h]$.

Besides the unrestricted 2-jumping relation, we also define other four different types of limited 2-jumping relations, which are the main subject of this paper. The presented limitation restricts the jumping direction of jumps, moreover, it restricts each jump separately; which in total gives four different possible variants of such limitation. As we show further, these restrictions severely and uniquely impact the automata's behavior and thus the families of accepted languages.

(1) M makes a *right-right* 2*-jump* from $\zeta_1\zeta_2$ to $\vartheta_1\vartheta_2$ according to h, symbolically written as

$$\zeta_1\zeta_2 \ {}_{\blacktriangleright\blacktriangleright}\!\curvearrowright \ \vartheta_1\vartheta_2 \ \ [h]$$

if and only if $\zeta_1 \ {}_{\blacktriangleright}\!\curvearrowright \vartheta_1$ $[h]$ and $\zeta_2 \ {}_{\blacktriangleright}\!\curvearrowright \vartheta_2$ $[h]$;
(2) M makes a *right-left* 2*-jump* from $\zeta_1\zeta_2$ to $\vartheta_1\vartheta_2$ according to h, symbolically written as

$$\zeta_1\zeta_2 \ {}_{\blacktriangleright\blacktriangleleft}\!\curvearrowright \ \vartheta_1\vartheta_2 \ \ [h]$$

if and only if $\zeta_1 \ {}_{\blacktriangleright}\!\curvearrowright \vartheta_1$ $[h]$ and $\zeta_2 \ {}_{\blacktriangleleft}\!\curvearrowright \vartheta_2$ $[h]$;
(3) M makes a *left-right* 2*-jump* from $\zeta_1\zeta_2$ to $\vartheta_1\vartheta_2$ according to h, symbolically written as

$$\zeta_1\zeta_2 \ {}_{\blacktriangleleft\blacktriangleright}\!\curvearrowright \ \vartheta_1\vartheta_2 \ \ [h]$$

if and only if $\zeta_1 \ {}_{\blacktriangleleft}\!\curvearrowright \vartheta_1$ $[h]$ and $\zeta_2 \ {}_{\blacktriangleright}\!\curvearrowright \vartheta_2$ $[h]$; and
(4) M makes a *left-left* 2*-jump* from $\zeta_1\zeta_2$ to $\vartheta_1\vartheta_2$ according to h, symbolically written as

$$\zeta_1\zeta_2 \ {}_{\blacktriangleleft\blacktriangleleft}\!\curvearrowright \ \vartheta_1\vartheta_2 \ \ [h]$$

if and only if $\zeta_1 \ {}_{\blacktriangleleft}\!\curvearrowright \vartheta_1$ $[h]$ and $\zeta_2 \ {}_{\blacktriangleleft}\!\curvearrowright \vartheta_2$ $[h]$.

Let o be any of the jumping direct relations introduced above. In the standard way, extend o to o^m, $m \geq 0$; o^+; and o^*. Let $M = (Q, \Sigma, R, s, F)$ be a GJFA. To express that M only performs jumps according to o, write M_o. If o is one of the relations $\curvearrowright$, ${}_{\blacktriangleright}\!\curvearrowright$, ${}_{\blacktriangleleft}\!\curvearrowright$, ${}_{\blacklozenge}\!\curvearrowright$, set

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

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

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

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

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

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

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

Furthermore, set $\mathscr{L}_2 = \mathscr{L}_{{}_{\blacklozenge\blacklozenge}\!\curvearrowright} \cup \mathscr{L}_{{}_{\blacktriangleright\blacktriangleright}\!\curvearrowright} \cup \mathscr{L}_{{}_{\blacktriangleright\blacktriangleleft}\!\curvearrowright} \cup \mathscr{L}_{{}_{\blacktriangleleft\blacktriangleright}\!\curvearrowright} \cup \mathscr{L}_{{}_{\blacktriangleleft\blacktriangleleft}\!\curvearrowright}$.

Lastly, we introduce a subfamily of the family of regular languages essential to the study of the generative power of GJFAs that perform right-left and left-right 2-jumps.

Definition 2.1 *Let $L_{m,n}$ be a* simply-expandable language*, a SEL for short, over an alphabet Σ if it can be written as follows. Let m and n be positive integers; then,*

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

For the sake of clarity, let us note that, in the previous definition, $u_{h,k}$ and v_h are fixed strings that only vary for different values of h.

Throughout the rest of this paper, the remaining language families under discussion are denoted in the following way. **FIN**, **REG**, **LIN**, **CF**, **CS**, and **SEL** denote the families of finite languages, regular languages, linear languages, context-free languages, context-sensitive languages, and SELs, respectively.

3. Results

This section studies the generative power of GJFAs making their computational steps by unrestricted, right-left, left-right, right-right, and left-left 2-jumps.

3.1. On the unrestricted 2-jumping relation

Example 3.1 *Consider the GJFA*

$$M_{\blacklozenge\blacklozenge\curvearrowright} = (\{s, f\}, \Sigma, R, s, \{f\}),$$

where $\Sigma = \{a, b, c\}$ and R consists of the rules (s, ab, f) and (f, c, f). Starting from s, M has to read two times some ab, entering the final state f; then, M can arbitrarily many times read two times some c. Consequently, if we work with the unrestricted 2-jumps, the input must always contain two separate strings ab and the symbols c can be anywhere around these two strings. Therefore, the accepted language is

$$L(M_{\blacklozenge\blacklozenge\curvearrowright}) = \{c^k abc^m abc^n \mid k + m + n \text{ is an even integer},\ k, m, n \geq 0\}.$$

Lemma 3.2 *For every language $L \in \mathscr{L}_2$, there is no $x \in L$ such that $|x|$ is an odd number.*

Proof. By the definition of 2-jumps, any GJFA that uses 2-jumps always performs two single jumps simultaneously and they both follow the same rule, therefore, there is no way how to read an odd number of symbols from the input string. □

Lemma 3.3 *There is no GJFA $M_{\curvearrowright}$ that accepts $\{c^k abc^m abc^n \mid k + m + n$ is an even integer, $k, m, n \geq 0\}$.*

Proof. We follow Lemma 19 from [6] which effectively shows that a GJFA $M_{\curvearrowright}$ can maintain a specific order of symbols only in the sole context of a rule. Therefore, by contradiction. Let

$K = \{c^k abc^m abc^n \mid k+m+n \text{ is an even integer, } k, m, n \geq 0\}$. Assume that there is a GJFA $M_{\curvearrowright}$ such that $L(M_{\curvearrowright}) = K$. If M uses two times a rule reading ab, then it can also accept input $aabb$ and clearly $aabb \notin K$. Consequently, M has to always read the whole sequence $abc^m ab$ with a single rule; however, number m is unbounded and thus there cannot be finitely many rules that cover all possibilities—a contradiction with the assumption that $L(M_{\curvearrowright}) = K$ exists. Therefore, there is no GJFA $M_{\curvearrowright}$ that accepts $\{c^k abc^m abc^n \mid k + m + n \text{ is an even integer, } k, m, n \geq 0\}$. □

Theorem 3.4 *$\mathscr{L}_{\curvearrowright}$ and $\mathscr{L}_{\blacklozenge\blacklozenge\curvearrowright}$ are incomparable.*

Proof. $\mathscr{L}_{\curvearrowright} \not\subseteq \mathscr{L}_{\blacklozenge\blacklozenge\curvearrowright}$ follows from **FIN** $\subset \mathscr{L}_{\curvearrowright}$ (see Theorem 18 in [6]) and Lemma 3.2. $\mathscr{L}_{\blacklozenge\blacklozenge\curvearrowright} \not\subseteq \mathscr{L}_{\curvearrowright}$ follows from Example 3.1 and Lemma 3.3. Moreover, observe that both $\mathscr{L}_{\curvearrowright}$ and $\mathscr{L}_{\blacklozenge\blacklozenge\curvearrowright}$ clearly contain simple finite language $\{aa\}$. □

3.2. On the right-left 2-jumping relation

Claim 3.5 *Let $M = (Q, \Sigma, R, s, F)$ be a GJFA; then, every $x \in L(M_{\blacktriangleright\blacktriangleleft\curvearrowright})$ can be written as $x = u_1 u_2 \dots u_n u_n \dots u_2 u_1$, where $n \in \mathbb{N}$, and $u_i \in \Sigma^*$, $1 \leq i \leq n$.*

Proof. Consider any GJFA $M_{\blacktriangleright\blacktriangleleft\curvearrowright} = (Q, \Sigma, R, s, F)$. Since we work with the right-left 2-jumps, the first jump can move only to the right, the second jump can move only to the left, and both jumps cannot cross each other. Observe that if the configuration of M is of the form $upvpw$, where $u, v, w \in \Sigma^*$, and $p \in Q$, then M cannot read the symbols in u and w anymore. Also, observe that this covers the situation when M starts to accept $x \in \Sigma^*$ from another configuration than sxs. Therefore, to read the whole input string, M has to start in the configuration sxs and it cannot jump over any symbols during the whole process. Consequently, since both jumps always follow the same rule, they have to read the same corresponding strings and, at the end, meet in the middle of the input string. Therefore, every $x \in L(M_{\blacktriangleright\blacktriangleleft\curvearrowright})$ can be surely written as $x = u_1 u_2 \dots u_n u_n \dots u_2 u_1$, where $n \in \mathbb{N}$, and $u_i \in \Sigma^*$, $1 \leq i \leq n$. □

Lemma 3.6 *For every GJFA M, there is a linear grammar G such that $L(M_{\blacktriangleright\blacktriangleleft\curvearrowright}) = L(G)$.*

Proof. Consider any GJFA $M_{\blacktriangleright\blacktriangleleft\curvearrowright} = (Q, \Sigma, R, s, F)$. Define the linear grammar

$$G = (Q, \Sigma, P, s),$$

where P is constructed in the following way:

1. For each $(p, y, q) \in R$, add $p \to yqy$ to P.
2. For each $p \in F$, add $p \to \varepsilon$ to P.

We follow Claim 3.5 and its proof. Let $p, q \in Q$, $f \in F$, and $y, u, v, w \in \Sigma^*$. Observe that every time M can make a 2-jump $pywyp \blacktriangleright\blacktriangleleft\curvearrowright qwq$ according to $(p, y, q) \in P$, G can also make the derivation step $upv \Rightarrow uyqyv$ according to $p \to yqy \in P$. Moreover, every time M is in a final state f, G can finish the string with $f \to \varepsilon \in P$. Finally, observe that G cannot do any other action, therefore, $L(M_{\blacktriangleright\blacktriangleleft\curvearrowright}) = L(G)$. □

Theorem 3.7 $\mathscr{L}_{\blacktriangleright\blacktriangleleft\curvearrowright} \subset \boldsymbol{LIN}$.

Proof. $\mathscr{L}_{\blacktriangleright\blacktriangleleft\curvearrowright} \subseteq \mathbf{LIN}$ follows from Lemma 3.6. $\mathbf{LIN} \not\subseteq \mathscr{L}_{\blacktriangleright\blacktriangleleft\curvearrowright}$ follows from Lemma 3.2. □

Claim 3.8 *There is a GJFA M such that $L(M_{\blacktriangleright\blacktriangleleft\curvearrowright}) = \{w \in \Sigma^* \mid w \text{ is an even palindrome}\}$.*

Proof. Consider an arbitrary alphabet Σ. Define the GJFA

$$M_{\blacktriangleright\blacktriangleleft\curvearrowright} = (\{f\}, \Sigma, R, f, \{f\})$$

where $R = \{(f, a, f) \mid a \in \Sigma\}$.

We follow Claim 3.5 and its proof, which shows that every $x \in L(M_{\blacktriangleright\blacktriangleleft\curvearrowright})$ can be written as $x = u_1u_2 \dots u_nu_n \dots u_2u_1$, where $n \in \mathbb{N}$, and $u_i \in \Sigma^*$, $1 \leq i \leq n$. Observe that we use only rules reading single symbols, thus we can even say that $u_i \in (\Sigma \cup \{\varepsilon\})$, $1 \leq i \leq n$, which, in fact, models the string pattern of the even palindrome. Moreover, we use only one sole state that can accept all symbols from Σ, therefore, $L(M_{\blacktriangleright\blacktriangleleft\curvearrowright}) = \{w \in \Sigma^* \mid w \text{ is an even palindrome}\}$. □

Lemma 3.9 *For every SEL $K_{m,n}$, there is a GJFA M such that $K_{m,n} = L(M_{\blacktriangleright\blacktriangleleft\curvearrowright})$.*

Proof. Let m and n be positive integers. Consider any SEL over an alphabet Σ,

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

Define the GJFA $M_{\blacktriangleright\blacktriangleleft\curvearrowright} = (Q, \Sigma, R, \langle s\rangle, F)$, where Q, R, and F are constructed in the following way:

1. Add $\langle s\rangle$ to Q.
2. Add $\langle h, k\rangle$ to Q, for all $1 \leq h \leq m$, $1 \leq k \leq n+1$.
3. Add $\langle h, n+1\rangle$ to F, for all $1 \leq h \leq m$.
4. Add $(\langle s\rangle, \varepsilon, \langle h, 1\rangle)$ to R, for all $1 \leq h \leq m$.
5. Add $(\langle h, k\rangle, u_{h,k}, \langle h, k+1\rangle)$ to R, for all $1 \leq h \leq m$, $1 \leq k \leq n$.
6. Add $(\langle h, n+1\rangle, v_h, \langle h, n+1\rangle)$ to R, for all $1 \leq h \leq m$.

We follow Claim 3.5 and its proof. Observe that M starts from $\langle s\rangle$ by jumping to an arbitrary state $\langle h, 1\rangle$, where $1 \leq h \leq m$. Then, the first jump consecutively reads $u_{h,1}u_{h,2} \dots u_{h,n}$, and the second jump consecutively reads $u_{h,n} \dots u_{h,2}u_{h,1}$, until M ends up in the final state $\langle h, n+1\rangle$. Here, both jumps can arbitrarily many times read v_h. As a result, M accepts $u_{h,1}u_{h,2} \dots u_{h,n}v_h^i v_h^i u_{h,n} \dots u_{h,2}u_{h,1}$, for all $1 \leq h \leq m$, where $i \geq 0$, $u_{h,k}, v_h \in \Sigma^*$, $1 \leq k \leq n$; therefore, $K_{m,n} = L(M_{\blacktriangleright\blacktriangleleft\curvearrowright})$. □

Lemma 3.10 *For every SEL $K_{m,n}$, there is a right-linear grammar G such that $K_{m,n} = L(G)$.*

Proof. Let m and n be positive integers. Consider any SEL over an alphabet Σ,

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

Define the right-linear grammar $G = (N, \Sigma, P, \langle s\rangle)$, where N and P are constructed in the following way:

1. Add $\langle s \rangle$ to N.
2. Add $\langle h, 1 \rangle$ and $\langle h, 2 \rangle$ to N, for all $1 \leq h \leq m$.
3. Add $\langle s \rangle \to \langle h, 1 \rangle$ to P, for all $1 \leq h \leq m$.
4. Add $\langle h, 1 \rangle \to u_{h,1}u_{h,2} \dots u_{h,n}\langle h, 2 \rangle$ to P, for all $1 \leq h \leq m$.
5. Add $\langle h, 2 \rangle \to v_n v_n \langle h, 2 \rangle$ to P, for all $1 \leq h \leq m$.
6. Add $\langle h, 2 \rangle \to u_{h,n} \dots u_{h,2}u_{h,1}$ to P, for all $1 \leq h \leq m$.

Observe that at the beginning, G has to change nonterminal $\langle s \rangle$ to an arbitrary nonterminal $\langle h, 1 \rangle$, where $1 \leq h \leq m$. Then, it generates $u_{h,1}u_{h,2} \dots u_{h,n}$ and nonterminal $\langle h, 2 \rangle$. Here, it can arbitrarily many times generate $v_n v_n$ and ultimately finish the generation with $u_{h,n} \dots u_{h,2}u_{h,1}$. As a result, G generates $u_{h,1}u_{h,2} \dots u_{h,n}(v_h v_h)^i u_{h,n} \dots u_{h,2}u_{h,1}$, for all $1 \leq h \leq m$, where $i \geq 0$, $u_{h,k}, v_h \in \Sigma^*$, $1 \leq k \leq n$, which is indistinguishable from $u_{h,1}u_{h,2} \dots u_{h,n}v_h^i v_h^i u_{h,n} \dots u_{h,2}u_{h,1}$; therefore, $K_{m,n} = L(G)$. □

Theorem 3.11 ***SEL*** $\subset$ ***REG***.

Proof. **SEL** $\subseteq$ **REG** follows from Lemma 3.10. **REG** $\not\subseteq$ **SEL** follows from Lemma 3.9 and Lemma 3.2. □

Theorem 3.12 ***SEL*** $\subset \mathscr{L}_{\blacktriangleright\blacktriangleleft\curvearrowright}$.

Proof. **SEL** $\subseteq \mathscr{L}_{\blacktriangleright\blacktriangleleft\curvearrowright}$ follows from Lemma 3.9. $\mathscr{L}_{\blacktriangleright\blacktriangleleft\curvearrowright} \not\subseteq$ **SEL** follows from Theorem 3.11 and Claim 3.8 because a subfamily of the family of regular languages surely cannot contain a non-trivial language of all even palindromes. □

Theorem 3.13 *The following pairs of language families are incomparable:*

(i) $\mathscr{L}_{\blacktriangleright\blacktriangleleft\curvearrowright}$ *and* ***REG****;*
(ii) $\mathscr{L}_{\blacktriangleright\blacktriangleleft\curvearrowright}$ *and* ***FIN****.*

Proof. $\mathscr{L}_{\blacktriangleright\blacktriangleleft\curvearrowright} \not\subseteq$ **REG** and $\mathscr{L}_{\blacktriangleright\blacktriangleleft\curvearrowright} \not\subseteq$ **FIN** follow from Claim 3.8, Theorem 3.11 , and Theorem 3.12. **REG** $\not\subseteq \mathscr{L}_{\blacktriangleright\blacktriangleleft\curvearrowright}$ and **FIN** $\not\subseteq \mathscr{L}_{\blacktriangleright\blacktriangleleft\curvearrowright}$ follow from Lemma 3.2. Moreover, observe that $\mathscr{L}_{\blacktriangleright\blacktriangleleft\curvearrowright}$ clearly contains regular language $\{a^{2n} \mid n \geq 0\}$ and finite language $\{aa\}$. □

Open Problem 3.14 $(\mathscr{L}_{\blacktriangleright\blacktriangleleft\curvearrowright} -$ ***SEL***$) \cap$ ***REG*** $= \emptyset$*?*

3.3. On the left-right 2-jumping relation

Claim 3.15 *Let $M = (Q, \Sigma, R, s, F)$ be a GJFA; then, every $x \in L(M_{\blacktriangleleft\blacktriangleright\curvearrowright})$ can be written as $x = u_n \dots u_2 u_1 u_1 u_2 \dots u_n$, where $n \in \mathbb{N}$, and $u_i \in \Sigma^*$, $1 \leq i \leq n$.*

Proof. Consider any GJFA $M_{\blacktriangleleft\blacktriangleright\curvearrowright} = (Q, \Sigma, R, s, F)$. Since we work with the left-right 2-jumps, the first jump can move only to the left and the second jump can move only to the right. Observe that if the configuration of M is of the form $upvpw$, where $u, v, w \in \Sigma^*$, and $p \in Q$, then M cannot read the symbols in v anymore. Also, observe that this covers the situation when M

starts to accept $x \in \Sigma^*$ from another configuration than $yssz$, where $y, z \in \Sigma^*$ such that $x = yz$. Therefore, to read the whole input string, M has to start in the configuration $yssz$ and it cannot jump over any symbols during the whole process. Consequently, since both jumps always follow the same rule, they have to read the same corresponding strings and ultimately finish at the ends of the input string. Therefore, every $x \in L(M_{\blacktriangleleft\blacktriangleright\curvearrowright})$ can be surely written as $x = u_n \dots u_2 u_1 u_1 u_2 \dots u_n$, where $n \in \mathbb{N}$, and $u_i \in \Sigma^*$, $1 \leq i \leq n$. □

Lemma 3.16 *For every GJFA M, there is a GJFA N such that $L(M_{\blacktriangleleft\blacktriangleright\curvearrowright}) = L(N_{\blacktriangleright\blacktriangleleft\curvearrowright})$.*

Proof. Consider any $GJFA$ $M_{\blacktriangleleft\blacktriangleright\curvearrowright} = (Q, \Sigma, R_1, s_1, F)$. Without a loss of generality, assume that $s_2 \notin Q$. Define the GJFA

$$N_{\blacktriangleright\blacktriangleleft\curvearrowright} = (Q \cup \{s_2\}, \Sigma, R_2, s_2, \{s_1\}),$$

where R_2 is constructed in the following way:

1. For each $(p, y, q) \in R_1$, add (q, y, p) to R_2.
2. For each $f \in F$, add (s_2, ε, f) to R_2.

Note that this construction resembles the well-known conversion technique for finite automata which creates a finite automaton that accepts the reversal of the original language. However, observe that in this case, the effect is quite different. We follow Claims 3.5 and 3.15. Consider any $x \in L(M_{\blacktriangleleft\blacktriangleright\curvearrowright})$. We can surely find $x = u_n \dots u_2 u_1 u_1 u_2 \dots u_n$, where $n \in \mathbb{N}$, and $u_i \in \Sigma^*$, $1 \leq i \leq n$, such that N reads $u_n \dots u_2 u_1$ and $u_1 u_2 \dots u_n$ in the reverse order. Moreover, in N, both jumps have their direction reversed, compared to jumps in M, and thus they start on the opposite ends of their parts, which is demonstrated in the mentioned claims. Consequently, if each jump in N reads its part reversely and from the opposite end, then, in fact, N reads the same $u_n \dots u_2 u_1 u_1 u_2 \dots u_n$ as M. Finally, N surely cannot accept anything new that is not accepted by M, therefore, $L(M_{\blacktriangleleft\blacktriangleright\curvearrowright}) = L(N_{\blacktriangleright\blacktriangleleft\curvearrowright})$. □

Lemma 3.17 *For every GJFA M, there is a GJFA N such that $L(M_{\blacktriangleright\blacktriangleleft\curvearrowright}) = L(N_{\blacktriangleleft\blacktriangleright\curvearrowright})$.*

Proof. Consider any $GJFA$ $M_{\blacktriangleright\blacktriangleleft\curvearrowright} = (Q, \Sigma, R_1, s_1, F)$. Without a loss of generality, assume that $s_2 \notin Q$. Define the GJFA

$$N_{\blacktriangleleft\blacktriangleright\curvearrowright} = (Q \cup \{s_2\}, \Sigma, R_2, s_2, \{s_1\}),$$

where R_2 is constructed in the following way:

1. For each $(p, y, q) \in R_1$, add (q, y, p) to R_2.
2. For each $f \in F$, add (s_2, ε, f) to R_2.

The reasoning here is exactly the same as in Lemma 3.16. □

Theorem 3.18 $\mathscr{L}_{\blacktriangleleft\blacktriangleright\curvearrowright} = \mathscr{L}_{\blacktriangleright\blacktriangleleft\curvearrowright}$.

Proof. $\mathscr{L}_{\blacktriangleleft\blacktriangleright\curvearrowright} \subseteq \mathscr{L}_{\blacktriangleright\blacktriangleleft\curvearrowright}$ follows from Lemma 3.16. $\mathscr{L}_{\blacktriangleright\blacktriangleleft\curvearrowright} \subseteq \mathscr{L}_{\blacktriangleleft\blacktriangleright\curvearrowright}$ follows from Lemma 3.17. □

Corollary 3.19 *The following relations between language families hold:*

(i) $\mathscr{L}_{\blacktriangleleft\blacktriangleright\curvearrowright} \subset \boldsymbol{LIN}$;
(ii) $\boldsymbol{SEL} \subset \mathscr{L}_{\blacktriangleleft\blacktriangleright\curvearrowright}$;
(iii) $\mathscr{L}_{\blacktriangleleft\blacktriangleright\curvearrowright}$ *and* $\boldsymbol{REG}$ *are incomparable;*
(iv) $\mathscr{L}_{\blacktriangleleft\blacktriangleright\curvearrowright}$ *and* $\boldsymbol{FIN}$ *are incomparable.*

Proof. These results directly follow from Theorems 3.7, 3.12, 3.13, and 3.18. □

Open Problem 3.20 $(\mathscr{L}_{\blacktriangleleft\blacktriangleright\curvearrowright} - \boldsymbol{SEL}) \cap \boldsymbol{REG} = \emptyset$*?*

The results concerning the generative power of GJFAs that perform right-left and left-right 2-jumps are summarized in Figure 1.

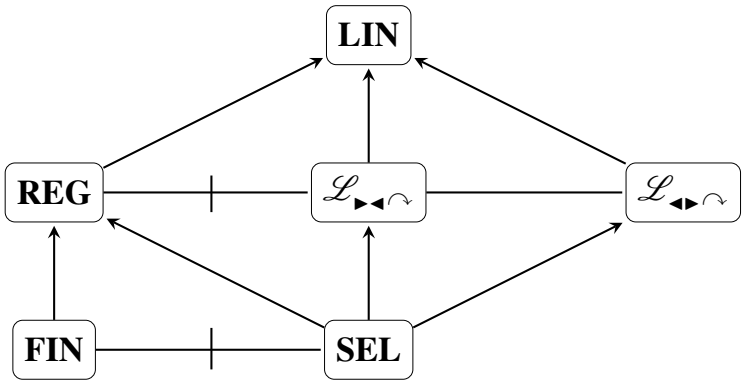

Figure 1: A hierarchy of language families closely related to the right-left and left-right 2-jumps is shown. If there is a line or an arrow from family X to family Y in the figure, then $X = Y$ or $X \subset Y$, respectively. A crossed line represents the incomparability between connected families.

3.4. On the right-right 2-jumping relation

Example 3.21 *Consider the GJFA*

$$M_{\blacktriangleright\blacktriangleright\curvearrowright} = (\{s, p, f\}, \Sigma, R, s, \{f\}),$$

where $\Sigma = \{a, b, c\}$ *and* R *consists of the rules* (s, ab, p) *and* (p, c, f)*. Starting from* s*,* M *has to read two times* ab *and two times* c*. Observe that if the first jump skips (jumps over) some symbols, then they cannot be ever read afterwards. However, the second jump is not so harshly restricted and can potentially skip some symbols which will be read later by the first jump. Therefore, the accepted language is*

$$L(M_{\blacktriangleright\blacktriangleright\curvearrowright}) = \{ababcc, abcabc\}.$$

Example 3.22 *Consider the GJFA*

$$M_{\blacktriangleright\blacktriangleright\frown} = (\{s, f\}, \Sigma, R, s, \{f\}),$$

where $\Sigma = \{a, b\}$ *and* R *consists of the rules* (s, b, f) *and* (f, a, f)*. Starting from* s*,* M *has to read two times* b *and then it can arbitrarily many times read two times* a*. Both jumps behave in the same way as in Example 3.21. Observe that when we consider no skipping of symbols, then* M *reads* ba^nba^n*,* $n \geq 0$*. Nevertheless, when we consider the skipping with the second jump, then the second* b *can also occur arbitrarily closer to the first* b*; until they are neighbors, and* M *reads* bba^{2n}*,* $n \geq 0$*. When combined together, the accepted language is*

$$L(M_{\blacktriangleright\blacktriangleright\frown}) = \{ba^nba^na^{2m} \mid n, m \geq 0\}.$$

Observe that this is clearly a non-regular context-free language.

Example 3.23 *Consider the GJFA*

$$M_{\blacktriangleright\blacktriangleright\frown} = (\{s, f\}, \Sigma, R, s, \{f\}),$$

where $\Sigma = \{a, b, c, d\}$ *and* $R = \{(s, y, f) \mid y \in \Sigma\} \cup \{(f, y, f) \mid y \in \Sigma\}$*. Starting from* s*,* M *has to read two times some symbol from* Σ *and then it can arbitrarily many times read two times any symbols from* Σ*. Again, both jumps behave in the same way as in Example 3.21. Consider the special case when the second jump consistently jumps over one symbol each time (except the last step) during the whole process. In such a case, the accepted strings can be written as* $u_1u'_1u_2u'_2 \dots u_nu'_n$*, where* $n \in \mathbb{N}$*,* $u_i, u'_i \in \Sigma$*,* $u_i = u'_i$*,* $1 \leq i \leq n$*. Observe that symbols without primes are read by the first jump, and symbols with primes are read by the second jump. Moreover, such strings can be surely generated by a right-linear grammar. Nevertheless, now consider no special case. Observe that, in the accepted strings, symbols with primes can be arbitrarily shifted to the right over symbols without primes, this creates a more complex structure, due to* $u_i = u'_i$*, with multiple crossed agreements. Lastly, consider the other border case with no skipping of any symbols at all. Then, the accepted strings can be written as* ww*, where* $w \in \Sigma^+$*. Such strings represent the reduplication phenomenon—the well-known example of non-context-free languages (see Chapter 3.1 in [7]). As a result, due to the unbound number of crossed agreements, we can safely state that* $L(M_{\blacktriangleright\blacktriangleright\frown})$ *is a non-context-free language.*

This statement can be formally proved by contradiction. Assume that $L(M_{\blacktriangleright\blacktriangleright\frown})$ *is a context-free language. The family of context-free languages is closed under intersection with regular sets. Let* $K = L(M_{\blacktriangleright\blacktriangleright\frown}) \cap ab^+c^+dab^+c^+d$*. Consider the previous description. Observe that this selects strings where* $u_1 = a$ *and* $u'_n = d$*. Since there are only exactly two symbols* a *and two symbols* d *in each selected string, we know where precisely both jumps start and end. And since the second jump starts after the position where the first jump ends, we also know that this, in fact, follows the special border case of behavior with no skipping of any symbols at all. Consequently,* $K = \{ab^nc^mdab^nc^md \mid n, m \geq 1\}$*. However,* K *is clearly a non-context-free language (see Chapter 3.1 in [7])—a contradiction with the assumption that* $L(M_{\blacktriangleright\blacktriangleright\frown})$ *is a context-free language. Therefore,* $L(M_{\blacktriangleright\blacktriangleright\frown})$ *is a non-context-free language.*

Theorem 3.24 $\mathscr{L}_{\blacktriangleright\blacktriangleright\curvearrowright} \subset \boldsymbol{CS}$.

Proof. Clearly, any GJFA $M_{\blacktriangleright\blacktriangleright\curvearrowright}$ can be simulated by linear bounded automata, so $\mathscr{L}_{\blacktriangleright\blacktriangleright\curvearrowright} \subseteq$ **CS**. **CS** $\not\subseteq \mathscr{L}_{\blacktriangleright\blacktriangleright\curvearrowright}$ follows from Lemma 3.2. □

Lemma 3.25 *Let $n \in \mathbb{N}$. For every GJFA M, where for every $x \in L(M_{\blacktriangleright\blacktriangleright\curvearrowright})$ holds either $|x| \leq n$ or $alph(x) = 1$, there is a right-linear grammar G such that $L(M_{\blacktriangleright\blacktriangleright\curvearrowright}) = L(G)$.*

Proof. Let $n \in \mathbb{N}$. Consider any GJFA $M_{\blacktriangleright\blacktriangleright\curvearrowright}$, where for every $x \in L(M_{\blacktriangleright\blacktriangleright\curvearrowright})$ holds either $|x| \leq n$ or $alph(x) = 1$. Define the right-linear grammar G in the following way. Observe that the number of x for which holds $|x| \leq n$ must be finite, therefore, for each such x, we can create a separate rule that generates x in G. On the other hand, the number of x for which holds $alph(x) = 1$ can be infinite, however, every such x is defined by the finite number of rules in M. And we can surely convert these rules (p, y, q) from M into rules in G in such a way that they generate y^2 and simulate the state transitions of M. Consequently, since the position of symbols here is ultimately irrelevant, these rules properly simulate results of 2-jumps in M. Therefore, $L(M_{\blacktriangleright\blacktriangleright\curvearrowright}) = L(G)$. □

Theorem 3.26 *The following pairs of language families are incomparable:*

(i) $\mathscr{L}_{\blacktriangleright\blacktriangleright\curvearrowright}$ *and* ***CF****;*
(ii) $\mathscr{L}_{\blacktriangleright\blacktriangleright\curvearrowright}$ *and* ***REG****;*
(iii) $\mathscr{L}_{\blacktriangleright\blacktriangleright\curvearrowright}$ *and* ***FIN****.*

Proof. $\mathscr{L}_{\blacktriangleright\blacktriangleright\curvearrowright} \not\subseteq$ **CF**, $\mathscr{L}_{\blacktriangleright\blacktriangleright\curvearrowright} \not\subseteq$ **REG**, and $\mathscr{L}_{\blacktriangleright\blacktriangleright\curvearrowright} \not\subseteq$ **FIN** follow from Example 3.23. **CF** $\not\subseteq \mathscr{L}_{\blacktriangleright\blacktriangleright\curvearrowright}$, **REG** $\not\subseteq \mathscr{L}_{\blacktriangleright\blacktriangleright\curvearrowright}$, and **FIN** $\not\subseteq \mathscr{L}_{\blacktriangleright\blacktriangleright\curvearrowright}$ follow from Lemma 3.2. Moreover, observe that $\mathscr{L}_{\blacktriangleright\blacktriangleright\curvearrowright}$ clearly contains the context-free language from Example 3.22, regular language $\{a^{2n} \mid n \geq 0\}$, and the finite language from Example 3.21. □

3.5. On the left-left 2-jumping relation

Example 3.27 *Consider the GJFA*

$$M_{\blacktriangleleft\blacktriangleleft\curvearrowright} = (\{s, p, f\}, \Sigma, R, s, \{f\}),$$

where $\Sigma = \{a, b, c\}$ and R consists of the rules (s, c, p) and (p, ab, f). Starting from s, M has to read two times c and two times ab. Observe that if the second jump skips some symbols, then they cannot be ever read afterwards. However, the first jump is not so harshly restricted and can potentially skip some symbols which will be read later by the second jump. Note that this precisely resembles the inverted behavior of the right-right 2-jumping relation. Therefore, the accepted language is

$$L(M_{\blacktriangleleft\blacktriangleleft\curvearrowright}) = \{ababcc, abacbc, abcabc\}.$$

Example 3.28 *Consider the GJFA*

$$M_{\blacktriangleleft\blacktriangleleft\curvearrowright} = (\{s, f\}, \Sigma, R, s, \{f\}),$$

where $\Sigma = \{a, b\}$ *and* R *consists of the rules* (s, a, s) *and* (s, b, f)*. Starting from* s*,* M *can arbitrarily many times read two times* a *and, at the end, it has to read two times* b*. Both jumps behave in the same way as in Example 3.27. Observe that when we consider no skipping of symbols, then* M *reads* ba^nba^n*,* $n \geq 0$*. Nevertheless, when we consider the skipping with the first jump, then the second* b *can also occur arbitrarily closer to the first* b*, since the first jump can now read symbols* a *also behind this second* b*. Consequently, the accepted language is*

$$L(M_{\blacktriangleleft\blacktriangleleft\curvearrowright}) = \{ba^nba^na^{2m} \mid n, m \geq 0\}.$$

Note that this is the same language as in Example 3.22.

Example 3.29 *Consider the GJFA*

$$M_{\blacktriangleleft\blacktriangleleft\curvearrowright} = (\{s, f\}, \Sigma, R, s, \{f\}),$$

where $\Sigma = \{a, b, c, d\}$ *and* $R = \{(s, y, f) \mid y \in \Sigma\} \cup \{(f, y, f) \mid y \in \Sigma\}$*. Starting from* s*,* M *has to read two times some symbol from* Σ *and then it can arbitrarily many times read two times any symbols from* Σ*. Both jumps behave in the same way as in Example 3.27 and the overall behavior tightly follows Example 3.23. Consider the special case when the first jump consistently jumps over one symbol each time (except the last step) during the whole process. In such a case, the accepted strings can be written as* $u'_n u_n \dots u'_2 u_2 u'_1 u_1$*, where* $n \in \mathbb{N}$*,* $u'_i, u_i \in \Sigma$*,* $u'_i = u_i$*,* $1 \leq i \leq n$*. Observe that symbols with primes are read by the first jump, and symbols without primes are read by the second jump. Now consider no special case. Observe that, in the accepted strings, symbols with primes can be arbitrarily shifted to the left over symbols without primes, which creates a more complex structure with multiple crossed agreements. And lastly, consider the other border case with no skipping of any symbols at all. Then, the accepted strings can be written as* ww*, where* $w \in \Sigma^+$*, which represents the reduplication phenomenon. As a result, due to the unbound number of crossed agreements, we can safely state that* $L(M_{\blacktriangleleft\blacktriangleleft\curvearrowright})$ *is a non-context-free language. This statement can be formally proved in the same way as in Example 3.23.*

Theorem 3.30 $\mathscr{L}_{\blacktriangleleft\blacktriangleleft\curvearrowright} \subset \boldsymbol{CS}$*.*

Proof. Clearly, any GJFA $M_{\blacktriangleleft\blacktriangleleft\curvearrowright}$ can be simulated by linear bounded automata, so $\mathscr{L}_{\blacktriangleleft\blacktriangleleft\curvearrowright} \subseteq$ **CS**. **CS** $\not\subseteq \mathscr{L}_{\blacktriangleleft\blacktriangleleft\curvearrowright}$ follows from Lemma 3.2. □

Lemma 3.31 *Let* $n \in \mathbb{N}$*. For every GJFA* M*, where for every* $x \in L(M_{\blacktriangleleft\blacktriangleleft\curvearrowright})$ *holds either* $|x| \leq n$ *or* $alph(x) = 1$*, there is a right-linear grammar* G *such that* $L(M_{\blacktriangleleft\blacktriangleleft\curvearrowright}) = L(G)$*.*

Proof. The reasoning here is exactly the same as in Lemma 3.25. □

Theorem 3.32 *The following pairs of language families are incomparable:*

(i) $\mathscr{L}_{\blacktriangleleft\blacktriangleleft\curvearrowright}$ *and* $\boldsymbol{CF}$*;*

(ii) $\mathscr{L}_{\blacktriangleleft\blacktriangleleft\curvearrowright}$ *and* ***REG****;*
(iii) $\mathscr{L}_{\blacktriangleleft\blacktriangleleft\curvearrowright}$ *and* ***FIN****.*

Proof. $\mathscr{L}_{\blacktriangleleft\blacktriangleleft\curvearrowright} \not\subseteq$ **CF**, $\mathscr{L}_{\blacktriangleleft\blacktriangleleft\curvearrowright} \not\subseteq$ **REG**, and $\mathscr{L}_{\blacktriangleleft\blacktriangleleft\curvearrowright} \not\subseteq$ **FIN** follow from Example 3.29. **CF** $\not\subseteq \mathscr{L}_{\blacktriangleleft\blacktriangleleft\curvearrowright}$, **REG** $\not\subseteq \mathscr{L}_{\blacktriangleleft\blacktriangleleft\curvearrowright}$, and **FIN** $\not\subseteq \mathscr{L}_{\blacktriangleleft\blacktriangleleft\curvearrowright}$ follow from Lemma 3.2. Moreover, observe that $\mathscr{L}_{\blacktriangleleft\blacktriangleleft\curvearrowright}$ clearly contains the context-free language from Example 3.28, regular language $\{a^{2n} \mid n \geq 0\}$, and the finite language from Example 3.27. □

Claim 3.33 *There is no GJFA* $M_{\blacktriangleright\blacktriangleright\curvearrowright}$ *that accepts* $\{ababcc, abacbc, abcabc\}$.

Proof. By contradiction. Let $K = \{ababcc, abacbc, abcabc\}$. Assume that there is a GJFA M such that $L(M_{\blacktriangleright\blacktriangleright\curvearrowright}) = K$. Observe that each string in K contains three pairs of symbols, therefore, to effectively read such a string, we need a maximum of three chained rules in M or less. (Note that additional rules reading ε do not affect results.) Moreover, due to the nature of strings in K, we need to consider only such chains of rules where, in the result, a precedes b, and b precedes c. Therefore, we can easily try all possibilities and calculate their resulting sets. Surely, $L(M_{\blacktriangleright\blacktriangleright\curvearrowright})$ must be a union of some of these sets:

(i) if M reads abc, the set is $\{abcabc\}$;
(ii) if M reads ab, and c, the set is $\{ababcc, abcabc\}$;
(iii) if M reads a, and bc, the set is $\{aabcbc, abacbc, abcabc\}$;
(iv) if M reads a, b, and c, the set is $\{aabbcc, ababcc, aabcbc, abacbc, abcabc\}$.

Clearly, no union of these sets can result in K—a contradiction with the assumption that $L(M_{\blacktriangleright\blacktriangleright\curvearrowright}) = K$ exists. Therefore, there is no GJFA $M_{\blacktriangleright\blacktriangleright\curvearrowright}$ that accepts $\{ababcc, abacbc, abcabc\}$. □

Claim 3.34 *There is no GJFA* $M_{\blacktriangleleft\blacktriangleleft\curvearrowright}$ *that accepts* $\{ababcc, abcabc\}$.

Proof. By contradiction. Let $K = \{ababcc, abcabc\}$. Assume that there is a GJFA M such that $L(M_{\blacktriangleleft\blacktriangleleft\curvearrowright}) = K$. Observe that each string in K contains three pairs of symbols, therefore, to effectively read such a string, we need a maximum of three chained rules in M or less. Moreover, due to the nature of strings in K, we need to consider only such chains of rules where, in the result, a precedes b, and b precedes c. Therefore, we can easily try all possibilities and calculate their resulting sets. Surely, $L(M_{\blacktriangleleft\blacktriangleleft\curvearrowright})$ must be a union of some of these sets:

(i) if M reads abc, the set is $\{abcabc\}$;
(ii) if M reads c, and ab, the set is $\{ababcc, abacbc, abcabc\}$;
(iii) if M reads bc, and a, the set is $\{aabcbc, abcabc\}$;
(iv) if M reads c, b, and a, the set is $\{aabbcc, aabcbc, ababcc, abacbc, abcabc\}$.

Clearly, no union of these sets can result in K—a contradiction with the assumption that $L(M_{\blacktriangleleft\blacktriangleleft\curvearrowright}) = K$ exists. Therefore, there is no GJFA $M_{\blacktriangleleft\blacktriangleleft\curvearrowright}$ that accepts $\{ababcc, abcabc\}$. □

Theorem 3.35 *$\mathscr{L}_{\blacktriangleright\blacktriangleright\curvearrowright}$ and $\mathscr{L}_{\blacktriangleleft\blacktriangleleft\curvearrowright}$ are incomparable.*

Proof. $\mathscr{L}_{\blacktriangleright\blacktriangleright\curvearrowright} \not\subseteq \mathscr{L}_{\blacktriangleleft\blacktriangleleft\curvearrowright}$ follows from Example 3.21 and Claim 3.34. $\mathscr{L}_{\blacktriangleleft\blacktriangleleft\curvearrowright} \not\subseteq \mathscr{L}_{\blacktriangleright\blacktriangleright\curvearrowright}$ follows from Example 3.27 and Claim 3.33. Moreover, observe that both $\mathscr{L}_{\blacktriangleright\blacktriangleright\curvearrowright}$ and $\mathscr{L}_{\blacktriangleleft\blacktriangleleft\curvearrowright}$ clearly contain the same language from Examples 3.22 and 3.28. □

The results concerning the generative power of GJFAs that perform right-right and left-left 2-jumps are summarized in Figure 2.

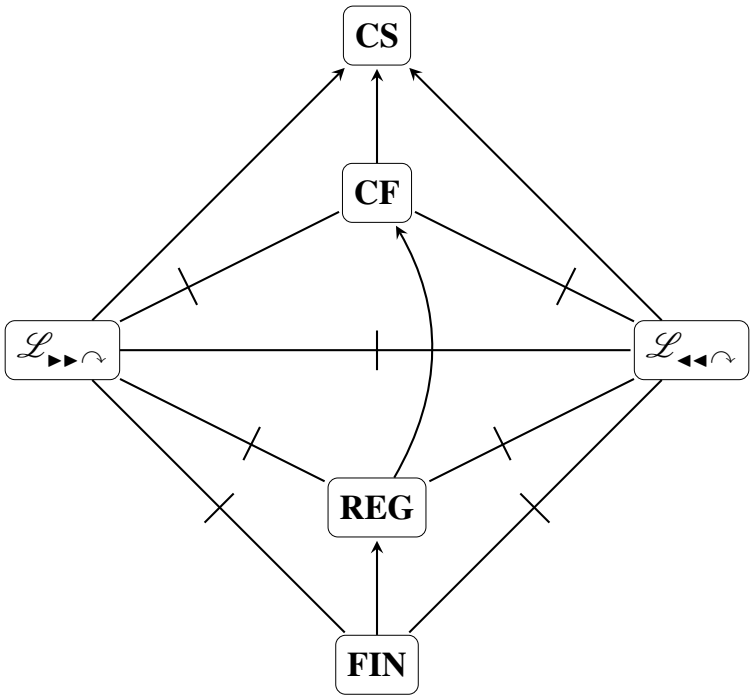

Figure 2: A hierarchy of language families closely related to the right-right and left-left 2-jumps is shown. If there is a line or an arrow from family X to family Y in the figure, then $X = Y$ or $X \subset Y$, respectively. A crossed line represents the incomparability between connected families.

4. Conclusion

We propose some future investigation areas concerning jumping finite automata that link several jumps together. Within the previous sections, we have already pointed out two specific open problems concerning right-left and left-right 2-jumps. This concluding section continues with other more general and broader suggestions.

(I.) Study closure properties of the newly defined jumping modes.

(II.) Investigate remaining possible variants of 2-jumps where the unrestricted single jumps and the restricted single jumps are combined together.

(III.) Extend the definition of 2-jumps to the general definition of n-jumps, where $n \in \mathbb{N}$. Can we find some interesting general results about these multi-jumps?

(IV.) Study relaxed versions of 2-jumps where the single jumps do not have to follow the same rule and where each single jump have its own state.

(V.) Use the newly defined jumping modes in jumping finite automata in which rules read single symbols rather than whole strings (JFAs—see [6]).

(VI.) In the same fashion as in finite automata, consider deterministic versions of GJFAs with the newly defined jumping modes.

Acknowledgment

This work was supported by The Ministry of Education, Youth and Sports of the Czech Republic from the National Programme of Sustainability (NPU II); project IT4Innovations excellence in science - LQ1602; the TAČR grant TE01020415; and the BUT grant FIT-S-14-2299. The authors thank all the anonymous referees for their useful comments and suggestions.

References

[1] H. CHIGAHARA, S. Z. FAZEKAS, A. YAMAMURA, One-way Jumping Finite Automata. In: *The 77th National Convention of IPSJ*. 2015.

[2] H. FERNAU, M. PARAMASIVAN, M. L. SCHMID, Jumping Finite Automata: Characterizations and Complexity. In: *Implementation and Application of Automata - 20th International Conference, CIAA*. LNCS 9223, Springer, 2015, 89–101.

[3] Z. KŘIVKA, A. MEDUNA, Jumping Grammars. *International Journal of Foundations of Computer Science* 26 (2015) 6, 709–731.

[4] R. KOCMAN, A. MEDUNA, On Parallel Versions of Jumping Finite Automata. In: *Proceedings of the 2015 Federated Conference on Software Development and Object Technologies*. 2015. (in print).

[5] A. MEDUNA, *Automata and Languages: Theory and Applications*. Springer, London, 2000.

[6] A. MEDUNA, P. ZEMEK, Jumping Finite Automata. *International Journal of Foundations of Computer Science* 23 (2012) 7, 1555–1578.

[7] G. ROZENBERG, A. SALOMAA, *Handbook of Formal Languages, Vol. 2: Linear Modeling: Background and Application*. Springer-Verlag, 1997.

[8] V. VOREL, On Basic Properties of Jumping Finite Automata. *CoRR* abs/1511.08396 (2015). `http://arxiv.org/abs/1511.08396`

[9] V. VOREL, Two Results on Discontinuous Input Processing. *CoRR* abs/1511.08642 (2015). `http://arxiv.org/abs/1511.08642`

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