RAIRO-Theor. Inf. Appl. 52 (2018) 185–199
https://doi.org/10.1051/ita/2018013

RAIRO - Theoretical Informatics and Applications
www.rairo-ita.org

ON DOUBLE-JUMPING FINITE AUTOMATA AND THEIR CLOSURE PROPERTIES

Radim Kocman*, Zbyněk Křivka and Alexander Meduna

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. In addition, many closure and non-closure properties of the resulting language families are established.

Mathematics Subject Classification. 68Q45, 68Q70.

Accepted November 21, 2018.

1. Introduction

At present, jumping versions of rewriting systems, such as grammars and automata, represent a vivid investigation area in language theory (see [1–5, 7, 8, 14, 15]). 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 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: 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

Keywords and phrases: Discontinuous and parallel tape reading, general jumping finite automata, even-length languages, left and right jumps.

Centre of Excellence IT4Innovations, Faculty of Information Technology, Brno University of Technology, Božetěchova 2, Brno, Czech Republic.

* Corresponding author: ikocman@fit.vutbr.cz

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. In addition, the paper establishes several closure and non-closure properties concerning the language families defined by jumping finite automata working in the three ways sketched above.

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 performed by general jumping finite automata. Section 3 demonstrates the mutual relation between the language families mentioned above. Closure and non-closure properties of these families are covered in Section 4. Section 5 closes our study by pointing out some remarks and suggestions for the future investigation.

2. Preliminaries and definitions

This paper assumes that the reader is familiar with the theory of automata and formal languages (see [6, 16]). For a set Q, $\mathrm{card}(Q)$ denotes the cardinality of Q, and 2^Q denotes the power set 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*. Set $V^+ = V^* - \{\varepsilon\}$; algebraically, V^+ is thus the free semigroup generated by V under the operation of concatenation. For $x \in V^*$, $|x|$ denotes the length of x, and $\mathrm{alph}(x)$ denotes the set of all symbols occurring in x; for instance, $\mathrm{alph}(0010) = \{0, 1\}$. For $a \in V$, $|x|_a$ denotes the number of occurrences of a in x. Let $x = a_1a_2 \dots a_n$, where $a_i \in V$ for all $i = 1, \dots, n$, for some $n \geq 0$ ($x = \varepsilon$ if and only if $n = 0$). The *mirror image* of x, denoted by $\mathrm{mi}(x)$, is defined as $\mathrm{mi}(x) = a_na_{n-1} \dots a_1$. For $K \subseteq V^*$, we define $\mathrm{mi}(K) = \{\mathrm{mi}(x) \mid x \in K\}$. For $x, y \in V^*$, the *shuffle* of x and y, denoted by $\mathrm{shuffle}(x, y)$, is defined as $\mathrm{shuffle}(x, y) = \{x_1y_1x_2y_2 \dots x_ny_n \mid x = x_1x_2 \dots x_n, y = y_1y_2 \dots y_n, x_i, y_i \in V^*, 1 \leq i \leq n, n \geq 1\}$. For $K_1, K_2 \subseteq V^*$, $\mathrm{shuffle}(K_1, K_2) = \{z \mid z \in \mathrm{shuffle}(x, y),\ x \in K_1,\ y \in K_2\}$. Let X and Y be sets; we call X and Y to be *incomparable* if $X \not\subseteq Y$, $Y \not\subseteq X$, and $X \cap Y \neq \emptyset$. Let $\mathscr{L}$ be a language family. We say that $\mathscr{L}$ is *closed under endmarking* if and only if for every $L \in \mathscr{L}$, where $L \subseteq V^*$, for some alphabet V, $\# \notin V$ implies that $L\{\#\} \in \mathscr{L}$. We also say that $\mathscr{L}$ is *closed under endmarking on both sides* if and only if the previous implies that $\{\#\}L\{\#\} \in \mathscr{L}$.

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 rules 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 *lazy finite automaton* (see Sect. 2.6.2 in [16]), an *LFA* 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 $F \subseteq Q$ is a set of final states. If $(p, y, q) \in R$ implies that $|y| \leq 1$, then M is a *finite automaton*, an *FA* for short. If $(p, y, q) \in R$ and $x, y \in \Sigma^*$, then $pyx \Rightarrow qx$. In the standard manner, extend $\Rightarrow$ to $\Rightarrow^n$, where $n \geq 0$; then, based on $\Rightarrow^n$, define $\Rightarrow^+$ and $\Rightarrow^*$. The *language accepted by* M, denoted by $L(M)$, is defined as $L(M) = \{w \in \Sigma^* \mid sw \Rightarrow^* f, f \in F\}$.

A *general jumping finite automaton* (see [8]), a *GJFA* for short, is a quintuple $M = (Q, \Sigma, R, s, F)$, where Q, Σ, R, s, and F are defined as in an LFA. 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 $h\colon(p, y, q)$, and instead of $h\colon(p, y, q) \in R$, we simply write $h \in R$. A *configuration* of M is any string in $\Sigma^*Q\Sigma^*$. The binary *jumping relation*, symbolically denoted by ${}_{\blacklozenge}\!\curvearrowright$, over $\Sigma^*Q\Sigma^*$, is defined as follows. Let $x, z, x', z' \in \Sigma^*$ such that $xz = x'z'$ and $h\colon(p, y, q) \in R$; then, M makes a *jump* from $xpyz$ to $x'qz'$, symbolically written as $xpyz\ {}_{\blacklozenge}\!\curvearrowright x'qz'$ $[h]$. When the specification of the rule h is immaterial, we can omit $[h]$.

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, this paper considers three possible types of single jumps that can be used in this new double-jumping mode. Besides the unrestricted single jump ${}_{\blacklozenge}\!\curvearrowright$ from the original definition, we also define and use two restricted single jumps with limited movement. The definition of restricted jumps is modified from the original paper [8] in order to get a more consistent behavior. The restricted single jumps now read strings from the configuration on the specific side of the state depending on the actual direction of their jumping.

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

We combine two single jumps into the unrestricted and also four types of restricted 2-jumping relations. The restricted relations are the main subject of this paper, and we show in the next sections that these restrictions severely and uniquely impact the behavior of the automaton and thus also the families of accepted languages and their closure properties.

Let X denote the set of all configurations of M. A 2-*configuration* of M is any string in XX. Let X^2 denote the set of all 2-configurations of M. For brevity, let $t_1t_2 \in \{\blacklozenge\blacklozenge, \blacktriangleright\blacktriangleright, \blacktriangleright\blacktriangleleft, \blacktriangleleft\blacktriangleright, \blacktriangleleft\blacktriangleleft\}$ such that $t_1, t_2 \in \{\blacklozenge, \blacktriangleright, \blacktriangleleft\}$. The binary t_1t_2 2-*jumping relation*, symbolically denoted by ${}_{t_1t_2}\!\curvearrowright$, over X^2, is defined as follows. Let $\zeta_1\zeta_2, \vartheta_1\vartheta_2 \in X^2$, where $\zeta_1, \zeta_2, \vartheta_1, \vartheta_2 \in X$, and $h \in R$; then, M makes a t_1t_2 2-*jump* from $\zeta_1\zeta_2$ to $\vartheta_1\vartheta_2$ according to h, symbolically written as $\zeta_1\zeta_2\ {}_{t_1t_2}\!\curvearrowright\ \vartheta_1\vartheta_2$ $[h]$ if and only if $\zeta_1\ {}_{t_1}\!\curvearrowright \vartheta_1$ $[h]$ and $\zeta_2\ {}_{t_2}\!\curvearrowright \vartheta_2$ $[h]$. Depending on the specific type of jumps $\blacklozenge\blacklozenge$, $\blacktriangleright\blacktriangleright$, $\blacktriangleright\blacktriangleleft$, $\blacktriangleleft\blacktriangleright$, $\blacktriangleleft\blacktriangleleft$, we use the following naming: unrestricted, right–right, right–left, left-right, left–left 2-jumping relation (or 2-jump), respectively.

Let o be any of the jumping direct relations introduced above. In the standard way, extend o to o^m, $m \geq 0$; o^+; and o^*. To express that M only performs jumps according to o, write M_o. If o is one of the relations ${}_{\blacklozenge}\!\curvearrowright$, ${}_{\blacktriangleright}\!\curvearrowright$, ${}_{\blacktriangleleft}\!\curvearrowright$, 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 $\mathscr{L}_{\blacklozenge\blacklozenge\curvearrowright} = \{L(M_{\blacklozenge\blacklozenge\curvearrowright}) \mid M$ 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 define an auxiliary subfamily of the family of regular languages that will be useful to the study of the accepting power of GJFAs that perform right–left and left–right 2-jumps. In Section 3.2, it helps us to describe the regular portions of the appropriate language families.

Definition 2.1. Let $L_{m,n}$ be a *simply-expandable language* (SEL) over an alphabet Σ if it can be written as follows. Let m and n be positive integers; then,
$L_{m,n} = \bigcup_{h=1}^{m} \left\{u_{h,1}u_{h,2}\ldots u_{h,n}v_h^i v_h^i u_{h,n}\ldots u_{h,2}u_{h,1} \mid i \geq 0,\ u_{h,k}, v_h \in \Sigma^*,\ 1 \leq k \leq n\right\}.$

For the sake of clarity, let us note that, in the previous definition, v_h and all $u_{h,k}$ 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. Furthermore, $\mathbf{FIN}_{\text{even}}$,

$\mathbf{REG}_{\text{even}}$, $\mathbf{LIN}_{\text{even}}$, $\mathbf{CF}_{\text{even}}$, and $\mathbf{CS}_{\text{even}}$ denote their specific subfamilies that contain only languages with even-length strings.

3. General results

This section studies the accepting 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 $M_{\blacklozenge\blacklozenge\frown} = (\{s, f\}, \Sigma, R, s, \{f\})$, 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 $L(M_{\blacklozenge\blacklozenge\frown}) = \{c^k abc^m abc^n \mid k+m+n \text{ is an even integer},\ k, m, n \geq 0\}$.

Lemma 3.2. *For every language $L \in \mathscr{L}_2$, there is no $x \in L$ such that $|x|$ is an odd number; furthermore, there is no symbol a for which $|x|_a$ 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 $M_{\blacklozenge\frown}$ 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 [8] which effectively shows that a GJFA $M_{\blacklozenge\frown}$ 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 $M_{\blacklozenge\frown}$ such that $L(M_{\blacklozenge\frown}) = K$. 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 $L(M_{\blacklozenge\frown}) = K$ exists. Therefore, there is no GJFA $M_{\blacklozenge\frown}$ 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. *$\mathscr{L}_{\blacklozenge\frown}$ and $\mathscr{L}_{\blacklozenge\blacklozenge\frown}$ are incomparable.*

Proof. $\mathscr{L}_{\blacklozenge\frown} \not\subseteq \mathscr{L}_{\blacklozenge\blacklozenge\frown}$ follows from $\mathbf{FIN} \subset \mathscr{L}_{\blacklozenge\frown}$ (see Thm. 18 in [8]) and Lemma 3.2. $\mathscr{L}_{\blacklozenge\blacklozenge\frown} \not\subseteq \mathscr{L}_{\blacklozenge\frown}$ follows from Example 3.1 and Lemma 3.3. Moreover, both $\mathscr{L}_{\blacklozenge\frown}$ and $\mathscr{L}_{\blacklozenge\blacklozenge\frown}$ clearly contain the 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 $x \in L(M_{\blacktriangleright\blacktriangleleft\frown})$ can be written as $x = u_1u_2 \dots u_nu_n \dots u_2u_1$, where $n \in \mathbb{N}$, and $u_i \in \Sigma^*$, $1 \leq i \leq n$.*

Proof. Consider any GJFA $M_{\blacktriangleright\blacktriangleleft\frown} = (Q, \Sigma, R, s, F)$. 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 any other 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 $x \in L(M_{\blacktriangleright\blacktriangleleft\frown})$ can be surely written as $x = u_1u_2 \dots u_nu_n \dots u_2u_1$, where $n \in \mathbb{N}$, and $u_i \in \Sigma^*$, $1 \leq i \leq n$. □

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

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, $L(M_{\blacktriangleright\blacktriangleleft\curvearrowright}) = L(G)$. $\square$

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

Proof. $\mathscr{L}_{\blacktriangleright\blacktriangleleft\curvearrowright} \subseteq \mathbf{LIN}_{\text{even}}$ follows from Lemma 3.6 and the structure of its proof. $\mathbf{LIN}_{\text{even}} \not\subseteq \mathscr{L}_{\blacktriangleright\blacktriangleleft\curvearrowright}$ follows from Lemma 3.2. $\square$

Claim 3.8. *There is a GJFA M such that $L(M_{\blacktriangleright\blacktriangleleft\curvearrowright}) = \{w \in \Sigma^* \mid w$ 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 $x = u_1 u_2 \dots u_n u_n \dots u_2 u_1$, where $n \in \mathbb{N}$, and $u_i \in \Sigma^*$, $1 \le i \le n$. Observe that we use only rules reading single symbols, thus we can even say that $u_i \in (\Sigma \cup \{\varepsilon\})$, $1 \le i \le n$, 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, $L(M_{\blacktriangleright\blacktriangleleft\curvearrowright}) = \{w \in \Sigma^* \mid w$ is an even palindrome$\}$. $\square$

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,
$K_{m,n} = \bigcup_{h=1}^{m} \left\{u_{h,1} u_{h,2} \dots u_{h,n} v_h^i v_h^i u_{h,n} \dots u_{h,2} u_{h,1} \mid i \ge 0,\ u_{h,k}, v_h \in \Sigma^*,\ 1 \le k \le n\right\}$.
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 $1 \le h \le m$, $1 \le k \le n+1$.
(3) Add $\langle h, n+1 \rangle$ to F, for all $1 \le h \le m$.
(4) Add $(\langle s \rangle, \varepsilon, \langle h, 1 \rangle)$ to R, for all $1 \le h \le m$.
(5) Add $(\langle h, k \rangle, u_{h,k}, \langle h, k+1 \rangle)$ to R, for all $1 \le h \le m$, $1 \le k \le n$.
(6) Add $(\langle h, n+1 \rangle, v_h, \langle h, n+1 \rangle)$ to R, for all $1 \le h \le m$.

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 $1 \le h \le m$. 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 $1 \le h \le m$, where $i \ge 0$, $u_{h,k}, v_h \in \Sigma^*$, $1 \le k \le n$; therefore, $K_{m,n} = L(M_{\blacktriangleright\blacktriangleleft\curvearrowright})$. $\square$

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,
$K_{m,n} = \bigcup_{h=1}^{m} \left\{u_{h,1} u_{h,2} \dots u_{h,n} v_h^i v_h^i u_{h,n} \dots u_{h,2} u_{h,1} \mid i \ge 0,\ u_{h,k}, v_h \in \Sigma^*,\ 1 \le k \le n\right\}$.
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 $1 \le h \le m$.
(3) Add $\langle s \rangle \to \langle h, 1 \rangle$ to P, for all $1 \le h \le m$.
(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 $1 \le h \le m$.
(5) Add $\langle h, 2 \rangle \to v_n v_n \langle h, 2 \rangle$ to P, for all $1 \le h \le m$.
(6) Add $\langle h, 2 \rangle \to u_{h,n} \dots u_{h,2} u_{h,1}$ to P, for all $1 \le h \le m$.

Observe that at the beginning, G has to change nonterminal $\langle s\rangle$ to an arbitrary nonterminal $\langle h,1\rangle$, where $1 \le h \le m$. 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 $1 \le h \le m$, where $i \ge 0$, $u_{h,k}, v_h \in \Sigma^*$, $1 \le k \le n$, 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, $K_{m,n} = L(G)$. $\square$

Theorem 3.11. *$\mathbf{SEL} \subset \mathbf{REG}_{even}$.*

Proof. $\mathbf{SEL} \subseteq \mathbf{REG}_{\mathrm{even}}$ follows from Lemma 3.10 and the structure of its proof. $\mathbf{REG}_{\mathrm{even}} \not\subseteq \mathbf{SEL}$ follows from Lemmas 3.9 and 3.2. $\square$

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

Proof. $\mathbf{SEL} \subseteq \mathscr{L}_{\blacktriangleright\blacktriangleleft\curvearrowright}$ follows from Lemma 3.9. $\mathscr{L}_{\blacktriangleright\blacktriangleleft\curvearrowright} \not\subseteq \mathbf{SEL}$ 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. $\square$

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

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

Proof. $\mathscr{L}_{\blacktriangleright\blacktriangleleft\curvearrowright} \not\subseteq \mathbf{REG}$ ($\mathbf{REG}_{\mathrm{even}}$) and $\mathscr{L}_{\blacktriangleright\blacktriangleleft\curvearrowright} \not\subseteq \mathbf{FIN}$ ($\mathbf{FIN}_{\mathrm{even}}$) follow from Claim 3.8, Theorem 3.11, and Theorem 3.12 (and Lem. 3.2). $\mathbf{REG}$ ($\mathbf{REG}_{\mathrm{even}}$) $\not\subseteq \mathscr{L}_{\blacktriangleright\blacktriangleleft\curvearrowright}$ and $\mathbf{FIN}$ ($\mathbf{FIN}_{\mathrm{even}}$) $\not\subseteq \mathscr{L}_{\blacktriangleright\blacktriangleleft\curvearrowright}$ follow from Lemma 3.2. Moreover, $\mathscr{L}_{\blacktriangleright\blacktriangleleft\curvearrowright}$ clearly contains the regular language $\{a^{2n} \mid n \ge 0\}$ and finite language $\{aa\}$. $\square$

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

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 $u_i \in \Sigma^*$, $1 \le i \le n$.*

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 any other 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 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 written as $x = u_n \dots u_2 u_1 u_1 u_2 \dots u_n$, where $n \in \mathbb{N}$, and $u_i \in \Sigma^*$, $1 \le i \le n$. $\square$

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, 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 $u_i \in \Sigma^*$, $1 \le i \le n$, 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 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. Thus, $L(M_{\blacktriangleleft\blacktriangleright\curvearrowright}) = L(N_{\blacktriangleright\blacktriangleleft\curvearrowright})$. $\square$

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. The construction and reasoning is exactly the same as in Lemma 3.16. $\square$

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. $\square$

Other properties of this language family thus coincide with Section 3.2.

3.4. On the right–right 2-jumping relation

Example 3.19. 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 3.20. Consider the GJFA $M_{\blacktriangleright\blacktriangleright\curvearrowright} = (\{s, f\}, \Sigma, R, s, \{f\})$, 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 3.19. Observe that when we consider no skipping of symbols, then M reads $ba^n ba^n$, $n \geq 0$. 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 $L(M_{\blacktriangleright\blacktriangleright\curvearrowright}) = \{ba^n ba^n a^{2m} \mid n, m \geq 0\}$. Observe that this is clearly a non-regular context-free language.

Example 3.21. Consider the GJFA $M_{\blacktriangleright\blacktriangleright\curvearrowright} = (\{s, f\}, \Sigma, R, s, \{f\})$, 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. Again, both jumps behave in the same way as in Example 3.19. 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 $u_1 u_1' u_2 u_2' \dots u_n u_n'$, where $n \in \mathbb{N}$, $u_i, u_i' \in \Sigma$, $u_i = u_i'$, $1 \leq i \leq n$. Observe that the symbols without primes are read by the first jump, and the 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, the 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 Chap. 3.1 in [13]). As a result, due to the unbound number of crossed agreements, we can safely state that $L(M_{\blacktriangleright\blacktriangleright\curvearrowright})$ is a non-context-free language.

This statement can be formally proven by contradiction. Assume that $L(M_{\blacktriangleright\blacktriangleright\curvearrowright})$ is a context-free language. The family of context-free languages is closed under intersection with regular sets. Let $K = L(M_{\blacktriangleright\blacktriangleright\curvearrowright}) \cap ab^+c^+dab^+c^+d$. 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, $K = \{ab^n c^m dab^n c^m d \mid n, m \geq 1\}$. However, K is clearly a non-context-free language (see Chap. 3.1 in [13]) – a contradiction with the assumption that $L(M_{\blacktriangleright\blacktriangleright\curvearrowright})$ is a context-free language. Therefore, $L(M_{\blacktriangleright\blacktriangleright\curvearrowright})$ is a non-context-free language.

Theorem 3.22. $\mathscr{L}_{\blacktriangleright\blacktriangleright\curvearrowright} \subset \boldsymbol{CS}_{even}$.

Proof. Clearly, any GJFA $M_{\blacktriangleright\blacktriangleright\curvearrowright}$ can be simulated by linear bounded automata, so $\mathscr{L}_{\blacktriangleright\blacktriangleright\curvearrowright} \subseteq \mathbf{CS}$. Due to Lemma 3.2, we can safely exclude all languages containing odd-length strings. $\mathbf{CS}_{\text{even}} \not\subseteq \mathscr{L}_{\blacktriangleleft\blacktriangleleft\curvearrowright}$ also follows from Lemma 3.2. □

Lemma 3.23. *Let $n \in \mathbb{N}$, and let M be any GJFA. Furthermore, let every $x \in L(M_{\blacktriangleright\blacktriangleright\curvearrowright})$ satisfy either $|x| \leq n$ or $\operatorname{alph}(x) = 1$. Then, there exists 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 every $x \in L(M_{\blacktriangleright\blacktriangleright\curvearrowright})$ satisfy either $|x| \leq n$ or $\operatorname{alph}(x) = 1$. 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 $\operatorname{alph}(x) = 1$ 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 3.24. *The following pairs of language families are incomparable:*

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

Proof. $\mathscr{L}_{\blacktriangleright\blacktriangleright\curvearrowright} \not\subseteq \mathbf{CF}$ ($\mathbf{CF}_{\text{even}}$), $\mathscr{L}_{\blacktriangleright\blacktriangleright\curvearrowright} \not\subseteq \mathbf{REG}$ ($\mathbf{REG}_{\text{even}}$), and $\mathscr{L}_{\blacktriangleright\blacktriangleright\curvearrowright} \not\subseteq \mathbf{FIN}$ ($\mathbf{FIN}_{\text{even}}$) follow from Example 3.21. $\mathbf{CF}$ ($\mathbf{CF}_{\text{even}}$) $\not\subseteq \mathscr{L}_{\blacktriangleright\blacktriangleright\curvearrowright}$, $\mathbf{REG}$ ($\mathbf{REG}_{\text{even}}$) $\not\subseteq \mathscr{L}_{\blacktriangleright\blacktriangleright\curvearrowright}$, and $\mathbf{FIN}$ ($\mathbf{FIN}_{\text{even}}$) $\not\subseteq \mathscr{L}_{\blacktriangleright\blacktriangleright\curvearrowright}$ follow from Lemma 3.2. Moreover, observe that $\mathscr{L}_{\blacktriangleright\blacktriangleright\curvearrowright}$ clearly contains the context-free language from Example 3.20, regular language $\{a^{2n} \mid n \geq 0\}$, and finite language from Example 3.19. □

3.5. On the left–left 2-jumping relation

Example 3.25. 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. As a result, the language is $L(M_{\blacktriangleleft\blacktriangleleft\curvearrowright}) = \{ababcc, abacbc, abcabc\}$.

Example 3.26. 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 3.25. 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 3.20.

Example 3.27. 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 3.25, and the overall behavior tightly follows Example 3.21. In the special case where the first jump consistently jumps over one symbol each time (except the last step) during the whole process, the accepted strings can be written as $u'_n u_n \ldots u'_2 u_2 u'_1 u_1$, where $n \in \mathbb{N}$, $u'_i, u_i \in \Sigma$, $u'_i = u_i$, $1 \leq i \leq n$. The symbols with primes are read by the first jump, and the symbols without primes are read by the second jump. With no special case, the symbols with primes can be arbitrarily shifted to the left over the symbols without primes, which creates a more complex structure with multiple crossed agreements and ultimately also the structure of the reduplication phenomenon. As a result, we can safely state that $L(M_{\blacktriangleleft\blacktriangleleft\frown})$ is a non-context-free language, and this statement can be formally proven in the same way as in Example 3.21.

Theorem 3.28. $\mathscr{L}_{\blacktriangleleft\blacktriangleleft\frown} \subset \boldsymbol{CS}_{even}$.

Proof. The reasoning is identical to Theorem 3.22. $\square$

Lemma 3.29. *Let $n \in \mathbb{N}$, and let M be any GJFA. Furthermore, let every $x \in L(M_{\blacktriangleleft\blacktriangleleft\frown})$ satisfy either $|x| \leq n$ or $\mathrm{alph}(x) = 1$. Then, there exists a right-linear grammar G such that $L(M_{\blacktriangleleft\blacktriangleleft\frown}) = L(G)$.*

Proof. The reasoning is exactly the same as in Lemma 3.23. $\square$

Theorem 3.30. *The following pairs of language families are incomparable:*

(i) $\mathscr{L}_{\blacktriangleleft\blacktriangleleft\frown}$ *and* $\boldsymbol{CF}$ ($\boldsymbol{CF}_{even}$);
(ii) $\mathscr{L}_{\blacktriangleleft\blacktriangleleft\frown}$ *and* $\boldsymbol{REG}$ ($\boldsymbol{REG}_{even}$);
(iii) $\mathscr{L}_{\blacktriangleleft\blacktriangleleft\frown}$ *and* $\boldsymbol{FIN}$ ($\boldsymbol{FIN}_{even}$).

Proof. $\mathscr{L}_{\blacktriangleleft\blacktriangleleft\frown} \nsubseteq \mathbf{CF}$ ($\mathbf{CF}_{\text{even}}$), $\mathscr{L}_{\blacktriangleleft\blacktriangleleft\frown} \nsubseteq \mathbf{REG}$ ($\mathbf{REG}_{\text{even}}$), and $\mathscr{L}_{\blacktriangleleft\blacktriangleleft\frown} \nsubseteq \mathbf{FIN}$ ($\mathbf{FIN}_{\text{even}}$) follow from Example 3.27. $\mathbf{CF}$ ($\mathbf{CF}_{\text{even}}$) $\nsubseteq \mathscr{L}_{\blacktriangleleft\blacktriangleleft\frown}$, $\mathbf{REG}$ ($\mathbf{REG}_{\text{even}}$) $\nsubseteq \mathscr{L}_{\blacktriangleleft\blacktriangleleft\frown}$, and $\mathbf{FIN}$ ($\mathbf{FIN}_{\text{even}}$) $\nsubseteq \mathscr{L}_{\blacktriangleleft\blacktriangleleft\frown}$ follow from Lemma 3.2. Moreover, $\mathscr{L}_{\blacktriangleleft\blacktriangleleft\frown}$ contains the context-free language from Example 3.26, regular language $\{a^{2n} \mid n \geq 0\}$, and finite language from Example 3.25. $\square$

Claim 3.31. *There is no GJFA $M_{\blacktriangleright\blacktriangleright\frown}$ that accepts $\{ababcc, abacbc, abcabc\}$.*

Proof. By contradiction. Let $K = \{ababcc, abacbc, abcabc\}$. Assume that there is a GJFA M such that $L(M_{\blacktriangleright\blacktriangleright\frown}) = K$. 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, $L(M_{\blacktriangleright\blacktriangleright\frown})$ 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 $L(M_{\blacktriangleright\blacktriangleright\frown}) = K$ exists. Therefore, there is no GJFA $M_{\blacktriangleright\blacktriangleright\frown}$ that accepts $\{ababcc, abacbc, abcabc\}$. $\square$

Claim 3.32. *There is no GJFA $M_{\blacktriangleleft\blacktriangleleft\frown}$ that accepts $\{ababcc, abcabc\}$.*

Proof. By contradiction. Let $K = \{ababcc, abcabc\}$. Assume that there is a GJFA M such that $L(M_{\blacktriangleleft\blacktriangleleft\frown}) = K$. By the same reasoning as in the proof of Claim 3.31, $L(M_{\blacktriangleleft\blacktriangleleft\frown})$ 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. Therefore, there is no GJFA $M_{\blacktriangleleft\blacktriangleleft\frown}$ that accepts $\{ababcc, abcabc\}$. $\square$

Theorem 3.33. $\mathscr{L}_{\blacktriangleright\blacktriangleright\frown}$ *and* $\mathscr{L}_{\blacktriangleleft\blacktriangleleft\frown}$ *are incomparable.*

Proof. $\mathscr{L}_{\blacktriangleright\blacktriangleright\frown} \nsubseteq \mathscr{L}_{\blacktriangleleft\blacktriangleleft\frown}$ follows from Example 3.19 and Claim 3.32. $\mathscr{L}_{\blacktriangleleft\blacktriangleleft\frown} \nsubseteq \mathscr{L}_{\blacktriangleright\blacktriangleright\frown}$ follows from Example 3.25 and Claim 3.31. Moreover, both $\mathscr{L}_{\blacktriangleright\blacktriangleright\frown}$ and $\mathscr{L}_{\blacktriangleleft\blacktriangleleft\frown}$ clearly contain the same language from Examples 3.20 and 3.26. $\square$

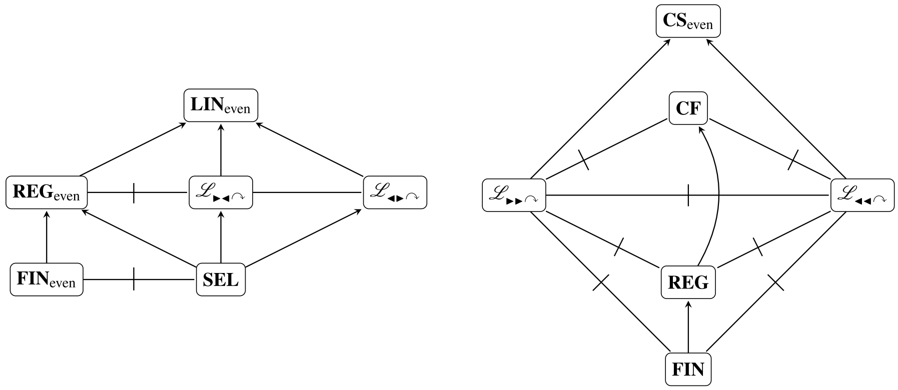

FIGURE 1. The hierarchies of closely related language families are shown. If there is a line or an arrow from family X to family Y in the figure, then $X = Y$ or $X \subset Y$, respectively. A crossed line represents the incomparability between connected families.

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

4. Closure properties

In this section, we show the closure properties of $\mathscr{L}_{\blacktriangleright\blacktriangleleft\curvearrowright}$, $\mathscr{L}_{\blacktriangleleft\blacktriangleright\curvearrowright}$, $\mathscr{L}_{\blacktriangleright\blacktriangleright\curvearrowright}$, and $\mathscr{L}_{\blacktriangleleft\blacktriangleleft\curvearrowright}$ under various operations. Recall that, by Theorem 3.18, $\mathscr{L}_{\blacktriangleright\blacktriangleleft\curvearrowright}$ and $\mathscr{L}_{\blacktriangleleft\blacktriangleright\curvearrowright}$ are equivalent, and so their closure properties coincide.

Theorem 4.1. *All $\mathscr{L}_{\blacktriangleright\blacktriangleleft\curvearrowright}$ ($\mathscr{L}_{\blacktriangleleft\blacktriangleright\curvearrowright}$), $\mathscr{L}_{\blacktriangleright\blacktriangleright\curvearrowright}$, and $\mathscr{L}_{\blacktriangleleft\blacktriangleleft\curvearrowright}$ are not closed under endmarking.*

Proof. This result directly follows from Lemma 3.2 – the inability to read an odd number of symbols from the input string. □

Theorem 4.2. *$\mathscr{L}_{\blacktriangleright\blacktriangleleft\curvearrowright}$ ($\mathscr{L}_{\blacktriangleleft\blacktriangleright\curvearrowright}$) is closed under endmarking on both sides.*

Proof. Consider any GJFA $M_{\blacktriangleright\blacktriangleleft\curvearrowright} = (Q, \Sigma, R, s, F)$. Without a loss of generality, assume that $s' \notin Q$ and $\# \notin \Sigma$. Define GJFA $N_{\blacktriangleright\blacktriangleleft\curvearrowright} = (Q \cup \{s'\}, \Sigma \cup \{\#\}, R \cup \{(s', \#, s)\}, s', F)$. Then, by Claim 3.5, every $x \in L(N_{\blacktriangleright\blacktriangleleft\curvearrowright})$ can be surely written as $x = \#u_2u_3 \ldots u_nu_n \ldots u_3u_2\#$, where $n \in \mathbb{N}$, and $u_i \in \Sigma^*$, $2 \leq i \leq n$. □

Theorem 4.3. *Both $\mathscr{L}_{\blacktriangleright\blacktriangleright\curvearrowright}$ and $\mathscr{L}_{\blacktriangleleft\blacktriangleleft\curvearrowright}$ are not closed under endmarking on both sides.*

Proof. Since both jumps always read the same strings in the same direction, they clearly cannot reliably define the endmarking on the opposite sides of the input string in the general case. □

Theorem 4.4. *All $\mathscr{L}_{\blacktriangleright\blacktriangleleft\curvearrowright}$ ($\mathscr{L}_{\blacktriangleleft\blacktriangleright\curvearrowright}$), $\mathscr{L}_{\blacktriangleright\blacktriangleright\curvearrowright}$, and $\mathscr{L}_{\blacktriangleleft\blacktriangleleft\curvearrowright}$ are not closed under concatenation.*

Proof. This can be easily proven by contradiction. Consider two simple languages $\{aa\}$ and $\{bb\}$, which clearly belong into $\mathscr{L}_{\blacktriangleright\blacktriangleleft\curvearrowright}$, $\mathscr{L}_{\blacktriangleright\blacktriangleright\curvearrowright}$, and $\mathscr{L}_{\blacktriangleleft\blacktriangleleft\curvearrowright}$. Assume that $\mathscr{L}_{\blacktriangleright\blacktriangleleft\curvearrowright}$, $\mathscr{L}_{\blacktriangleright\blacktriangleright\curvearrowright}$, and $\mathscr{L}_{\blacktriangleleft\blacktriangleleft\curvearrowright}$ are closed under concatenation. Therefore, the resulting language $\{aabb\}$ also have to belong into $\mathscr{L}_{\blacktriangleright\blacktriangleleft\curvearrowright}$, $\mathscr{L}_{\blacktriangleright\blacktriangleright\curvearrowright}$, and $\mathscr{L}_{\blacktriangleleft\blacktriangleleft\curvearrowright}$. However, such a language does not satisfy the string form for $\mathscr{L}_{\blacktriangleright\blacktriangleleft\curvearrowright}$ from Claim 3.5, and there is no GJFA $M_{\blacktriangleright\blacktriangleright\curvearrowright}$ or GJFA $N_{\blacktriangleleft\blacktriangleleft\curvearrowright}$ that can define such a language. Observe that M and N cannot accept $aabb$ with a single 2-jump, and that the rules for multiple 2-jumps define broader languages, *e.g.* $\{abab, aabb\}$. □

Theorem 4.5. *All $\mathscr{L}_{\blacktriangleright\blacktriangleleft\curvearrowright}$ ($\mathscr{L}_{\blacktriangleleft\blacktriangleright\curvearrowright}$), $\mathscr{L}_{\blacktriangleright\blacktriangleright\curvearrowright}$, and $\mathscr{L}_{\blacktriangleleft\blacktriangleleft\curvearrowright}$ are not closed under square.*

Proof. Consider language $L = \{aa, bb\}$, which clearly belongs into $\mathscr{L}_{\blacktriangleright\blacktriangleleft\curvearrowright}$, $\mathscr{L}_{\blacktriangleright\blacktriangleright\curvearrowright}$, and $\mathscr{L}_{\blacktriangleleft\blacktriangleleft\curvearrowright}$. Therefore, $L^2 = \{aaaa, aabb, bbaa, bbbb\}$ should also belong into these language families. However, observe string $aabb$, it causes the same problems as in the proof of Theorem 4.4. This string does not satisfy the string form for $\mathscr{L}_{\blacktriangleright\blacktriangleleft\curvearrowright}$ from

Claim 3.5. Moreover, there is no GJFA $M_{\blacktriangleright\blacktriangleright\curvearrowright}$ or GJFA $N_{\blacktriangleleft\blacktriangleleft\curvearrowright}$ that can simultaneously accept required string *aabb* and reject unwanted string *abab*. □

Theorem 4.6. *All $\mathscr{L}_{\blacktriangleright\blacktriangleleft\curvearrowright}$ ($\mathscr{L}_{\blacktriangleleft\blacktriangleright\curvearrowright}$), $\mathscr{L}_{\blacktriangleright\blacktriangleright\curvearrowright}$, and $\mathscr{L}_{\blacktriangleleft\blacktriangleleft\curvearrowright}$ are not closed under shuffle.*

Proof. Consider two simple languages $\{aa\}$ and $\{bb\}$, which clearly belong into $\mathscr{L}_{\blacktriangleright\blacktriangleleft\curvearrowright}$, $\mathscr{L}_{\blacktriangleright\blacktriangleright\curvearrowright}$, and $\mathscr{L}_{\blacktriangleleft\blacktriangleleft\curvearrowright}$. Therefore, the resulting language of their shuffle $\{aabb, abab, baab, abba, baba, bbaa\}$ should also belong into these language families. However, several strings from this language do not satisfy the string form for $\mathscr{L}_{\blacktriangleright\blacktriangleleft\curvearrowright}$ from Claim 3.5. Moreover, there is surely no GJFA $M_{\blacktriangleright\blacktriangleright\curvearrowright}$ or GJFA $N_{\blacktriangleleft\blacktriangleleft\curvearrowright}$ that can accept string *baab* or *abba*, since these strings do not contain two identical sequences of symbols that could be properly synchronously read. □

Theorem 4.7. *All $\mathscr{L}_{\blacktriangleright\blacktriangleleft\curvearrowright}$ ($\mathscr{L}_{\blacktriangleleft\blacktriangleright\curvearrowright}$), $\mathscr{L}_{\blacktriangleright\blacktriangleright\curvearrowright}$, and $\mathscr{L}_{\blacktriangleleft\blacktriangleleft\curvearrowright}$ are closed under union.*

Proof. Let o be one of the relations $\blacktriangleright\blacktriangleleft\curvearrowright$, $\blacktriangleright\blacktriangleright\curvearrowright$, and $\blacktriangleleft\blacktriangleleft\curvearrowright$; and $M_o = (Q_1, \Sigma_1, R_1, s_1, F_1)$, and $N_o = (Q_2, \Sigma_2, R_2, s_2, F_2)$ be two GJFAs. Without a loss of generality, assume that $Q_1 \cap Q_2 = \emptyset$ and $s \notin (Q_1 \cup Q_2)$. Define the GJFA $H_o = (Q_1 \cup Q_2 \cup \{s\}, \Sigma_1 \cup \Sigma_2, R_1 \cup R_2 \cup \{(s, \varepsilon, s_1), (s, \varepsilon, s_2)\}, s, F_1 \cup F_2)$. Observe that $L(H_o) = L(M_o) \cup L(N_o)$ holds in all modes. Indeed, the leading 2-jump only selects whether H_o enters M_o or N_o, and this leading 2-jump introduces no other new configuration to the configurations of M_o and N_o. □

Theorem 4.8. *All $\mathscr{L}_{\blacktriangleright\blacktriangleleft\curvearrowright}$ ($\mathscr{L}_{\blacktriangleleft\blacktriangleright\curvearrowright}$), $\mathscr{L}_{\blacktriangleright\blacktriangleright\curvearrowright}$, and $\mathscr{L}_{\blacktriangleleft\blacktriangleleft\curvearrowright}$ are not closed under complement.*

Proof. Consider Lemma 3.2 – that all 2-jumping modes can only accept even-length input strings. As a result, every complement has to contain at least all odd-length strings, and thus it cannot be defined by any 2-jumping mode. □

Theorem 4.9. *$\mathscr{L}_{\blacktriangleright\blacktriangleleft\curvearrowright}$ ($\mathscr{L}_{\blacktriangleleft\blacktriangleright\curvearrowright}$) is closed under intersection with regular languages.*

Proof. Consider any GJFA $M_{\blacktriangleright\blacktriangleleft\curvearrowright} = (Q_1, \Sigma, R_1, s_1, F_1)$ and FA $N = (Q_2, \Sigma, R_2, s_2, F_2)$. We can define a new GJFA $H_{\blacktriangleright\blacktriangleleft\curvearrowright} = (Q_3, \Sigma, R_3, s_3, F_3)$ that simulates both M and N in the same time, and that accepts the input string x if and only if both M and N also accept x. Note that the requirement of identical Σ does not affect the generality of the result. We are going to use two auxiliary functions that will help us with the construction of H. First, $\mathrm{Fw}(N, p, str)$ that accepts three parameters: N which is some FA, p which is some state of N, and str which is some string. This function returns the set of states in which N can end up if N is in state p and reads str. Second, $\mathrm{Bw}(N, p, str)$ that also accepts the same parameters: N which is some FA, p which is some state of N, and str which is some string. This function returns the set of states from which N reads str and ends in state p. We are not giving full details of these functions here since they only incorporate the well-known standard techniques for finite automata. With this, we construct Q_3, R_3, and F_3 in the following way:

(1) Add s_3 to Q_3.
(2) Add $\langle p, q, r\rangle$ to Q_3, for all $(p, q, r) \in Q_1 \times Q_2 \times Q_2$.
(3) Add $\langle p, q, q\rangle$ to F_3, for all $(p, q) \in F_1 \times Q_2$.
(4) Add $(s_3, \varepsilon, \langle s_1, s_2, f\rangle)$ to R_3, for all $f \in F_2$.
(5) For each $(p, a, q) \in R_1$ and $r_1, t_1 \in Q_2$, add $(\langle p, r_1, t_1\rangle, a, \langle q, r_2, t_2\rangle)$ to R_3, for all $(r_2, t_2) \in \mathrm{Fw}(N, r_1, a) \times \mathrm{Bw}(N, t_1, a)$.

Observe that H handles three distinct things in its states $\langle p, q, r\rangle$: p represents the original state of M, q simulates the first part of N in the classical forward way, and r simulates the second part of N in the backward way. At the beginning, H makes a 2-jump from the initial state s_3 into one of the states $\langle s_1, s_2, f\rangle$, where $f \in F_2$, and the main part of the simulation starts. In each following step, H can only make a 2-jump if the similar 2-jump is also in M and if N can read the same string as M from the both opposite sides with the current states. This part ends when there are no valid 2-jumps or when H reads the whole input string. If H processes the whole input string, we can recognize valid final state $\langle p, q, r\rangle$ in the following way: p has to be the original final state of M, and q must be the same as r so that the simulation of N from the two opposite sides can be connected in the middle. As a result, $L(H_{\blacktriangleright\blacktriangleleft\curvearrowright}) = L(M_{\blacktriangleright\blacktriangleleft\curvearrowright}) \cap L(N)$. □

Theorem 4.10. *$\mathscr{L}_{\blacktriangleright\blacktriangleleft\curvearrowright}$ ($\mathscr{L}_{\blacktriangleleft\blacktriangleright\curvearrowright}$) is closed under intersection.*

Proof. Consider any GJFA $M_{\blacktriangleright\blacktriangleleft\curvearrowright} = (Q_1, \Sigma, R_1, s_1, F_1)$ and GJFA $N_{\blacktriangleright\blacktriangleleft\curvearrowright} = (Q_2, \Sigma, R_2, s_2, F_2)$. We can define a new GJFA $H_{\blacktriangleright\blacktriangleleft\curvearrowright} = (Q, \Sigma, R, s, F)$ that simulates both M and N in the same time such that $L(H_{\blacktriangleright\blacktriangleleft\curvearrowright}) = L(M_{\blacktriangleright\blacktriangleleft\curvearrowright}) \cap L(N_{\blacktriangleright\blacktriangleleft\curvearrowright})$. To support the construction of Q and R, define $\Sigma^{\leq h} = \bigcup_{i=0}^{h} \Sigma^i$, and let k be the maximum length of the right-hand sides of the rules from $R_1 \cup R_2$. First, set Q to $\{\langle q_1, x, x', q_2, y, y'\rangle \mid q_1 \in Q_1,\ q_2 \in Q_2,\ x, x', y, y' \in \Sigma^{\leq 2k-1}\}$, F to $\{\langle f_1, \varepsilon, \varepsilon, f_2, \varepsilon, \varepsilon\rangle \mid f_1 \in F_1, f_2 \in F_2\}$, and $s = \langle s_1, \varepsilon, \varepsilon, s_2, \varepsilon, \varepsilon\rangle$. Then, we construct R in the following way:

(I) Add $(\langle p, x, x', q, y, y'\rangle, a, \langle p, xa, ax', q, ya, ay'\rangle)$ to R, for all $a \in \Sigma^{\leq k}$, $p \in Q_1$, $q \in Q_2$, and $x, x', y, y' \in \Sigma^{\leq 2k-1-|a|}$.

(II) For each $(p, a, p') \in R_1$, add $(\langle p, ax, x'a, q, y, y'\rangle, \varepsilon, \langle p', x, x', q, y, y'\rangle)$ to R, for all $x, x' \in \Sigma^{\leq 2k-1-|a|}$, $q \in Q_2$, and $y, y' \in \Sigma^{\leq 2k-1}$.

(III) For each $(q, b, q') \in R_2$, add $(\langle p, x, x', q, by, y'b\rangle, \varepsilon, \langle p, x, x', q', y, y'\rangle)$ to R, for all $p \in Q_1$, $x, x' \in \Sigma^{\leq 2k-1}$, and $y, y' \in \Sigma^{\leq 2k-1-|b|}$.

Observe that H stores six pieces of information in its compound states: (1) the state of M, (2) the buffered string (so called buffer) with up to $2k-1$ symbols read from the beginning of the input string to simulate the work of M on it, (3) the buffered string with up to $2k-1$ symbols read from the end of the input string to simulate the work of M on it, and pieces (4), (5), and (6) are analogous to (1), (2), and (3) but for N, respectively.

Next, by the same reasoning as in the proof of Claim 3.5, we can assume that M and N start from configurations s_1ws_1 and s_2ws_2, respectively, and neither of them can jump over any symbol during the reading. Using these assumptions, H simulates the work of M and N as follows. First, it reads by the rules from (I) a part of the input string and stores it in the buffers. Then, by the rules from (II) and (III), H processes the symbols from the buffers by the simulation of the rules from M and N. Whenever needed, H reads from the input string some additional symbols using the rules from (I). The input string is accepted by H if and only if the whole input string is read, all buffers are processed and emptied, and both (1) and (4) are final states of M and N, respectively.

To justify the maximum size of the buffers in (2), (3), (5), and (6), consider the situation when the simulation of M needs to read the input string by the words of length k, but the N's right-hand sides of the simulated rules alternate between 1 and k symbols. Then, we can observe a situation when a buffer contains $k-1$ symbols and we have to read k additional symbols from the input string before we can process the first (or the last) k symbols of the buffer. The question remains, however, whether we can reliably exclude some of these situations and possibly further decrease the size of the buffers in the states of H.

The rigorous proof of $L(H_{\blacktriangleright\blacktriangleleft\curvearrowright}) = L(M_{\blacktriangleright\blacktriangleleft\curvearrowright}) \cap L(N_{\blacktriangleright\blacktriangleleft\curvearrowright})$ is left to the reader. $\square$

Theorem 4.11. *Both $\mathscr{L}_{\blacktriangleright\blacktriangleright\curvearrowright}$ and $\mathscr{L}_{\blacktriangleleft\blacktriangleleft\curvearrowright}$ are not closed under intersection and intersection with regular languages.*

Proof. Consider two GJFAs:

$$\begin{aligned} M_{\blacktriangleright\blacktriangleright\curvearrowright} &= (\{s, r, p, f\}, \{a, b\}, \{(s, a, r), (r, bb, p), (p, a, f)\}, s, \{f\}); \\ L(M_{\blacktriangleright\blacktriangleright\curvearrowright}) &= \{abbaabba, abbabbaa, abababba, ababbbaa, aabbabba, aabbbbaa\}, \\ \text{and } N_{\blacktriangleright\blacktriangleright\curvearrowright} &= (\{s, r, p, f\}, \{a, b\}, \{(s, a, r), (r, b, p), (p, ba, f)\}, s, \{f\}); \\ L(N_{\blacktriangleright\blacktriangleright\curvearrowright}) &= \{abbaabba, abbababa, abababba, ababbaba, aabbabba, aabbbaba\}. \end{aligned}$$

The intersection $L_\cap = L(M_{\blacktriangleright\blacktriangleright\curvearrowright}) \cap L(N_{\blacktriangleright\blacktriangleright\curvearrowright}) = \{abbaabba, abababba, aabbabba\}$ should also belong into $\mathscr{L}_{\blacktriangleright\blacktriangleright\curvearrowright}$. However, consider the simplest GJFA $P_{\blacktriangleright\blacktriangleright\curvearrowright}$ that can accept string $aabbabba$; it surely has to start with reading two times only one symbol a, then it can read two times bb together, and then it finishes by reading two times symbol a. However, this is exactly the behavior of $M_{\blacktriangleright\blacktriangleright\curvearrowright}$, and we see that $L(M_{\blacktriangleright\blacktriangleright\curvearrowright})$ is a proper superset of

$L_\cap$. Therefore, there cannot be any GJFA $H_{\blacktriangleright\blacktriangleright\frown}$ that defines $L_\cap$. Trivially, both $L(M_{\blacktriangleright\blacktriangleright\frown})$ and $L(N_{\blacktriangleright\blacktriangleright\frown})$ are also regular languages. The similar proof for $\mathscr{L}_{\blacktriangleleft\blacktriangleleft\frown}$ is left to the reader. $\square$

Theorem 4.12. *$\mathscr{L}_{\blacktriangleright\blacktriangleleft\frown}$ ($\mathscr{L}_{\blacktriangleleft\blacktriangleright\frown}$) is closed under mirror image.*

Proof. Consider any GJFA $M_{\blacktriangleright\blacktriangleleft\frown} = (Q, \Sigma, R_1, s, F)$. Define the GJFA $N_{\blacktriangleright\blacktriangleleft\frown} = (Q, \Sigma, R_2, s, F)$, where R_2 is constructed in the following way. For each $(p, a, q) \in R_1$, add $(p, \mathrm{mi}(a), q)$ to R_2. Note that by Claim 3.5 and its proof, every $x \in L(M_{\blacktriangleright\blacktriangleleft\frown})$ can be written as $x = u_1u_2 \dots u_nu_n \dots u_2u_1$, where $n \in \mathbb{N}$, and $u_i \in \Sigma^*$, $1 \le i \le n$; and where each u_i represents string a from a certain rule. Observe that each x almost resembles an even palindrome. We just need to resolve the individual parts $|u_i| > 1$ for which the palindrome statement does not hold. Nevertheless, observe that if we simply mirror each u_i individually, it will create the mirror image of the whole x. As a result, $L(N_{\blacktriangleright\blacktriangleleft\frown})$ is a mirror image of $L(M_{\blacktriangleright\blacktriangleleft\frown})$. $\square$

Theorem 4.13. *Both $\mathscr{L}_{\blacktriangleright\blacktriangleright\frown}$ and $\mathscr{L}_{\blacktriangleleft\blacktriangleleft\frown}$ are not closed under mirror image.*

Proof. Consider language $K = \{ababcc, abcabc\}$, which is accepted by the GJFA $M_{\blacktriangleright\blacktriangleright\frown} = (\{s, r, f\}, \{a, b, c\}, \{(s, ab, r), (r, c, f)\}, s, \{f\})$. Therefore, the mirror language $K_{mi} = \{ccbaba, cbacba\}$ should also belong into $\mathscr{L}_{\blacktriangleright\blacktriangleright\frown}$. However, consider the simplest GJFA $N_{\blacktriangleright\blacktriangleright\frown}$ that can accept string $ccbaba$; it surely has to start with reading two times only symbol c, then it can read two times ba together. Even in such a case $L(N_{\blacktriangleright\blacktriangleright\frown}) = \{ccbaba, cbcaba, cbacba\}$; which is a proper superset of K_{mi}. Therefore, there cannot be any GJFA $H_{\blacktriangleright\blacktriangleright\frown}$ that defines K_{mi}. The similar proof for $\mathscr{L}_{\blacktriangleleft\blacktriangleleft\frown}$ is left to the reader. $\square$

Theorem 4.14. *All $\mathscr{L}_{\blacktriangleright\blacktriangleleft\frown}$ ($\mathscr{L}_{\blacktriangleleft\blacktriangleright\frown}$), $\mathscr{L}_{\blacktriangleright\blacktriangleright\frown}$, and $\mathscr{L}_{\blacktriangleleft\blacktriangleleft\frown}$ are not closed under finite substitution.*

Proof. Consider language $L = \{a^{2n} \mid n \ge 0\}$, which clearly belongs into $\mathscr{L}_{\blacktriangleright\blacktriangleleft\frown}$, $\mathscr{L}_{\blacktriangleright\blacktriangleright\frown}$, and $\mathscr{L}_{\blacktriangleleft\blacktriangleleft\frown}$. Define the finite substitution $\varphi : \{a\}^* \to 2^{\{a\}^*}$ as $\varphi(a) = \{\varepsilon, a\}$. Observe that $\varphi(L)$ contains odd-length strings. However, in consequence of Lemma 3.2, we know that no 2-jumping mode can accept such strings. $\square$

Theorem 4.15. *$\mathscr{L}_{\blacktriangleright\blacktriangleleft\frown}$ ($\mathscr{L}_{\blacktriangleleft\blacktriangleright\frown}$) is closed under homomorphism and ε-free homomorphism.*

Proof. Consider any GJFA $M_{\blacktriangleright\blacktriangleleft\frown} = (Q, \Sigma, R_1, s, F)$ and arbitrary homomorphism $\varphi : \Sigma^* \to \Delta^*$. Define the GJFA $N_{\blacktriangleright\blacktriangleleft\frown} = (Q, \Delta, R_2, s, F)$, where R_2 is constructed in the following way. For each $(p, a, q) \in R_1$, add $(p, \varphi(a), q)$ to R_2. Observe that by Claim 3.5 and its proof, every $x \in L(M_{\blacktriangleright\blacktriangleleft\frown})$ can be written as $x = u_1u_2 \dots u_nu_n \dots u_2u_1$, where $n \in \mathbb{N}$, and $u_i \in \Sigma^*$, $1 \le i \le n$; and where each u_i represents string a from a certain rule. Then, every $y \in L(N_{\blacktriangleright\blacktriangleleft\frown})$ can be surely written as $y = \varphi(u_1)\varphi(u_2) \dots \varphi(u_n)\varphi(u_n) \dots \varphi(u_2)\varphi(u_1)$, and clearly $\varphi(L(M_{\blacktriangleright\blacktriangleleft\frown})) = L(N_{\blacktriangleright\blacktriangleleft\frown})$. $\square$

Theorem 4.16. *Both $\mathscr{L}_{\blacktriangleright\blacktriangleright\frown}$ and $\mathscr{L}_{\blacktriangleleft\blacktriangleleft\frown}$ are not closed under homomorphism and ε-free homomorphism.*

Proof. Consider language $K = \{abab, aabb\}$, which is accepted by the GJFA $M_{\blacktriangleright\blacktriangleright\frown} = (\{s, r, f\}, \{a, b\}, \{(s, a, r), (r, b, f)\}, s, \{f\})$. Define the ε-free homomorphism $\varphi : \{a, b\}^+ \to \{a, b, c\}^+$ as $\varphi(a) = a$ and $\varphi(b) = bc$. By applying φ to K, we get $\varphi(K) = \{abcabc, aabcbc\}$. Consider the simplest GJFA $N_{\blacktriangleright\blacktriangleright\frown}$ that can accept string $aabcbc$; it surely has to start with reading two times only symbol a, then it can read two times bc together. However, even in such a case $L(N_{\blacktriangleright\blacktriangleright\frown}) = \{abcabc, abacbc, aabcbc\}$; which is a proper superset of $\varphi(K)$. Therefore, there cannot be any GJFA $H_{\blacktriangleright\blacktriangleright\frown}$ that defines $\varphi(K)$. Trivially, φ is also a general homomorphism. The similar proof for $\mathscr{L}_{\blacktriangleleft\blacktriangleleft\frown}$ is left to the reader. $\square$

Theorem 4.17. *All $\mathscr{L}_{\blacktriangleright\blacktriangleleft\frown}$ ($\mathscr{L}_{\blacktriangleleft\blacktriangleright\frown}$), $\mathscr{L}_{\blacktriangleright\blacktriangleright\frown}$, and $\mathscr{L}_{\blacktriangleleft\blacktriangleleft\frown}$ are not closed under inverse homomorphism.*

Proof. Consider language $L = \{aa\}$, which clearly belongs into $\mathscr{L}_{\blacktriangleright\blacktriangleleft\frown}$, $\mathscr{L}_{\blacktriangleright\blacktriangleright\frown}$, and $\mathscr{L}_{\blacktriangleleft\blacktriangleleft\frown}$. Define the homomorphism $\varphi : \{a\}^* \to \{a\}^*$ as $\varphi(a) = aa$. By applying φ^{-1} to L, we get $\varphi^{-1}(L) = \{a\}$. However, in consequence of Lemma 3.2, we know that no 2-jumping mode can define such a language. $\square$

The summary of closure properties of $\mathscr{L}_{\blacktriangleright\blacktriangleleft\frown}$, $\mathscr{L}_{\blacktriangleleft\blacktriangleright\frown}$, $\mathscr{L}_{\blacktriangleright\blacktriangleright\frown}$, and $\mathscr{L}_{\blacktriangleleft\blacktriangleleft\frown}$ is given in Figure 2, where $+$ marks closure, and $-$ marks non-closure.

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

FIGURE 2. Summary of closure properties.

5. REMARKS AND CONCLUSION

We would like to remark that the resulting behavior of right–left 2-jumps has proven to be very similar to the behaviors of 2-head finite automata accepting linear languages (see [11]) and $5' \to 3'$ sensing Watson–Crick finite automata (see [9, 10, 12]). Although these models differ in details, the general concept of their reading remains the same – all three mentioned models read simultaneously from the two different positions on the opposite sides of the input string. The main difference comes in the form of their rules. The other two models use more complex rules that allow them to read two different strings on their reading positions. Consequently, the resulting language families of these models differ from the language family defined by right–left 2-jumps. Nonetheless, the connection to Watson–Crick models shows that the concept of synchronized jumping could potentially find its use in the fields that study the correlations of several patterns such as biology or computer graphics. A further study of the combined model of jumping and $5' \to 3'$ Watson–Crick finite automata can be found in [4].

At the end, we propose some future investigation areas concerning jumping finite automata that link several jumps together. Within the previous sections, we have already pointed out one open problem concerning right–left (and left–right) 2-jumps (Open problem 3.14). This section continues with other more general suggestions.

(I.) Study decidability 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 [8]).
(VI.) In the same fashion as in finite automata, consider deterministic versions of GJFAs with the newly defined jumping modes.

Acknowledgements. 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-17-3964. The authors thank the anonymous referees for their useful comments and suggestions.

References

[1] H. Chigahara, S.Z. Fazekas and A. Yamamura, One-way jumping finite automata, in *The 77th Nat. Convention of IPSJ.* (2015).

[2] H. Fernau, M. Paramasivan, M.L. Schmid and V. Vorel, Characterization and complexity results on jumping finite automata. *Theor. Comput. Sci.* **679** (2017) 31–52.

[3] R. Kocman and A. Meduna, On parallel versions of jumping finite automata, in *Proc. of the 2015 Federated Conference on Software Development and Object Technologies (SDOT 2015).* Vol. 511 of Advances in Intelligent Systems and Computing. Springer, Cham (2016) 142–149.

[4] R. Kocman, B. Nagy, Z. Křivka and A. Meduna, A jumping $5' \to 3'$ watson-crick finite automata model, in *10th Workshop on Non-Classical Models of Automata and Applications (NCMA 2018)*, vol. 332. OCG books@ocg.at (2018) 117–132.

[5] Z. Křivka and A. Meduna, Jumping grammars. *Int. J. Found. Comput. Sci.* **26** (2015) 709–731.

[6] A. Meduna, Automata and Languages: Theory and Applications. Springer, London (2000).

[7] A. Meduna and O. Soukup, Jumping scattered context grammars. *Fundam. Inf.* **152** (2017) 51–86.

[8] A. Meduna and P. Zemek, Jumping finite automata. *Int. J. Found. Comput. Sci.* **23** (2012) 1555–1578.

[9] B. Nagy, On $5' \to 3'$ sensing watson-crick finite automata, in *DNA Computing: 13th International Meeting on DNA Computing (DNA13).* Vol. 4848 of *Lect. Notes Sci.* Springer, Berlin, Heidelberg (2008) 256–262.

[10] B. Nagy, $5' \to 3'$ sensing watson-crick finite automata, in Sequence and Genome Analysis II – Methods and Applications, edited by G. Fung. iConcept Press, Kowloon (2010) 39–56.

[11] B. Nagy, A class of 2-head finite automata for linear languages. *Triangle 8 (Lang.: Math. Approaches)* (2012) 89–99.

[12] B. Nagy, On a hierarchy of $5' \to 3'$ sensing watson-crick finite automata languages. *J. Logic Comput.* **23** (2013) 855–872.

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

[14] V. Vorel, Two results on discontinuous input processing, in Descriptional Complexity of Formal Systems (DCFS 2016). Vol. 9777 of *Lect. Notes Sci.* Springer, Cham (2016) 205–216.

[15] V. Vorel, On basic properties of jumping finite automata. *Int. J. Found. Comput. Sci.* **29** (2018) 1–15.

[16] D. Wood, Theory of Computation: A Primer. Addison-Wesley, Boston (1987).