Enhancement of Automata with Jumping Modes

Szilárd Zsolt Fazekas, Kaito Hoshi, Akihiro Yamamura

To cite this version:

Szilárd Zsolt Fazekas, Kaito Hoshi, Akihiro Yamamura. Enhancement of Automata with Jumping Modes. 25th International Workshop on Cellular Automata and Discrete Complex Systems (AUTOMATA), Jun 2019, Guadalajara, Mexico. pp.62-76, 10.1007/978-3-030-20981-0_5 . hal-02312614

HAL Id: hal-02312614

https://inria.hal.science/hal-02312614

Submitted on 11 Oct 2019

Enhancement of Automata with Jumping Modes

Szilárd Zsolt Fazekas[1]*, Kaito Hoshi[1], and Akihiro Yamamura[1]

Department of Mathematical Science and Electrical-Electronic-Computer Engineering, Akita University
1-1 Tegata Gakuen-machi, Akita 010-8502, JAPAN

Abstract. Recently, new types of non-sequential machine models have been introduced and studied, such as jumping automata and one-way jumping automata. We study the abilities and limitations of automata with these two jumping modes of tape heads with respect to how they affect the class of accepted languages. We give several methods to determine whether a language is accepted by a machine with jumping mode. We also consider relationships among the classes of languages defined by the new machines and their classical counterparts.

Keywords: Jumping mode · Jumping finite automata · Pushdown Automata · Pumping lemma · Context free language

1 Introduction

We study the ability of the jumping mode of tape heads to strengthen accepting power of automata. Automata have several characteristics; determinism of transition functions, ability of rewriting or erasing input words, memory devices like stacks and directions of tape heads to read inputs. Recently a mode of tape head movement has been introduced and examined with respect to how the class of languages accepted is affected ([1, 3–6, 8, 9, 11–13]). We study the abilities and limitations of the new mode of tape head move by comparing several machine models.

Jumping finite automata (JFA) were introduced as automata with a new mode of tape head in [12]. In the new mode, the tape head is allowed to jump - either left or right - over a part of the input word after reading a letter and continue processing from there. Once a letter in the input word is read, it cannot be reread again later. This implies that once a letter is read, it is erased. The tape head starts anywhere in the input word and it can move to either right or left side. It was shown that a language is accepted by jumping finite automata if and only if it is commutative and semilinear in [4, 5].

One-way jumping (deterministic) finite automata (OWJFA), a variant of jumping finite automata, were introduced and analyzed in [4]. They have another mode of tape head; the head moves in one direction only and starts at the beginning of the input word. It moves from left to right (and jumps over parts of

* Corresponding author

the input it cannot read) and when the tape head reaches the end of the input, it is returned to the beginning of the input and continues the computation until all the letters are read or the automaton is stuck in the sense that it can no longer read any letter of the remaining input. Several properties and characterization results were provided in [1]. The majority of decidability questions have been answered in [2], with the most notable exception being, when is the language accepted regular? We attempt to get closer to the answer by looking at the complements of OWJFA.

In this paper we consider deterministic or nondeterministic finite automata and pushdown automata in a uniform manner. However, we restrict our study to automata that do not rewrite the input letters and so we exclude linear bounded automata and Turing machines from consideration.

We recall notations of automata (see [7, 15]). We denote $\Sigma_\epsilon = \Sigma \cup \{\epsilon\}$, where ϵ is the empty word and $P(Q)$ stands for the power set of Q. A *nondeterministic finite automaton* (denoted by NFA) M is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$, where Q is set of states, Σ is finite set, $\delta : Q \times \Sigma \to P(Q)$ is a transition relation, q_0 is the initial state, F is set of accept state. In Section 3 we deviate from this definition by allowing the NFA to have multiple initial states, a change that is known not to affect the class of accepted languages in the classical case, but makes a difference in the alternative tape head modes. If δ is a mapping $Q \times \Sigma_\epsilon \to P(Q)$, then M is called $\epsilon-$NFA. M is called *deterministic* (denoted by a DFA) if (1) it is an ε-free NFA and (2) for $\forall p \in Q$ and $\forall a \in \Sigma$, there is no more than one $q \in Q$ such that $\delta(p, a) = q$.

A *nondeterministic pushdown automaton* (denoted by NPDA) M is a 6-tuple $(Q, \Sigma, \Gamma, \delta, q_0, F)$, where Q is finite set of states, Σ is finite set, Γ is a finite stack alphabet, $\delta : Q \times \Sigma_\epsilon \times \Gamma_\epsilon \to P(Q \times \Gamma_\epsilon)$ is the transition function, q_0 is the initial state, F is set of accept states, and \$ is a bottom marker of stack. M is called *deterministic* (denoted by DPDA) if it satisfies (1) For $\forall q \in Q, \forall a \in \Sigma \cup \{\epsilon\}, \forall b \in \Gamma$ we have $|\delta(q, a, b)| \leq 1$ and (2) For $\forall q \in Q, \forall a \in \Sigma, \forall b \in \Gamma$, we have $\delta(q, a, b) = \emptyset$ if $\delta(q, \epsilon, b) \neq \emptyset$. The language accepted by DPDA is defined as the set of inputs on which the automaton ends up in a final state (not the empty stack acceptance condition) after reading the whole input.

2 Modes of Tape Head Move

First we define modes of movement of the tape head to capture characteristics of both JFA and (R)OWJFA. Transitions between configurations of automata are considered to be rewriting of strings on state and input alphabets. We study three ways of rewriting configurations of automata; the standard mode, the nondeterministic jumping mode and the one-way jumping mode. The first one is the traditional way to rewrite configurations of automata as defined in [7] and [15]. The second and third one are introduced by [12] and [4], respectively. The three modes can be applied to any automata with deterministic/nondeterministic transition functions, with/without stacks, and with/without rewriting and erasing a letter in an input. Our objective with this paper is to extend the examination of how the three modes relate to each other with respect to the accepting power of automata, in particular, nondeterministic finite and pushdown automata.

2.1 Tape Head Modes

Suppose M is a (deterministic or nondeterministic) finite automaton.

Standard mode
A configuration of M is any string in $Q \times \Sigma^*$. A transition from configuration q_1aw to configuration q_2w, written as $q_1aw \rightarrow q_2w$, is possible when $q_2 \in \delta(q_1, a)$. In the standard manner, we extend $\rightarrow$ to $\rightarrow^m$, where $m \geq 0$. Let $\rightarrow^+$ and $\rightarrow^*$ denote the transitive closure and the transitive-reflexive closure of $\rightarrow$, respectively.

We define a (deterministic or nondeterministic) finite automaton with *standard mode* to be a rewriting system $(M, \rightarrow)$ based on $\rightarrow^*$. The language accepted by $(M, \rightarrow)$ (denoted by $L(M, \rightarrow)$) is defined to be $\{w \mid w \in \Sigma^*, sw \rightarrow^* f, f \in F\}$.

Nondeterministic jumping mode
A configuration of M is any string in $\Sigma^* \times Q \times \Sigma^*$, representing the part of the input to the left from the reading head, the state and the input to the right from the head. The binary jumping relation, symbolically denoted by $\curvearrowright$, over $\Sigma^* \times Q \times \Sigma^*$, is defined as follows. Let x, z, x', z' be strings in Σ^* such that $xz = x'z'$ and $q \in \delta(p, y)$; then, M makes a jump from $xpyz$ to $x'qz'$, symbolically written as $xpyz \curvearrowright x'qz'$. In the standard manner, we extend $\curvearrowright$ to $\curvearrowright^m$, where $m \geq 0$. Let $\curvearrowright^+$ and $\curvearrowright^*$ denote the transitive closure of $\curvearrowright$ and the transitive-reflexive closure of $\curvearrowright$, respectively.

We define a (deterministic or nondeterministic) finite automaton with *nondeterministic jumping mode* to be a rewriting system $(M, \curvearrowright)$ based on $\curvearrowright^*$. (see Fig. 1). The language accepted by $(M, \curvearrowright)$ (denoted by $L(M, \curvearrowright)$) is defined to be $\{uv \mid u, v \in \Sigma^*, usv \curvearrowright^* f, f \in F\}$. This jumping mode was introduced as a new automaton model called a JFA in [12].

One-way jumping mode
The *right one-way jumping relation* (denoted by $\circlearrowright$) between configurations from $Q\Sigma^*$, was defined in [4] as follows. Suppose that x and y belong to Σ^*, a belongs to Σ, p and q are states in Q and $q \in \delta(p, a)$. Then the right one-way jumping automaton M makes a jump from the configuration $pxay$ to the configuration qyx, symbolically written as $pxay \circlearrowright qyx$ if x belongs to $\{\Sigma \setminus \Sigma_p\}^*$ where $\Sigma_p = \{b \in \Sigma \mid \exists q \in Q \; s.t. \; q \in \delta(p, b)\}$ (see Fig. 2). In the standard manner, we extend $\circlearrowright$ to $\circlearrowright^m$, where $m \geq 0$. Let $\circlearrowright^*$ denote the transitive-reflexive closure of $\circlearrowright$.

We define a (deterministic or nondeterministic) finite automaton with *one-way jumping mode of tape head* to be a rewriting system $(M, \circlearrowright)$ based on $\circlearrowright^*$. The language accepted by $(M, \circlearrowright)$ (denoted by $L(M, \circlearrowright)$) is defined to be $\{w \mid w \in \Sigma^*, sw \circlearrowright f, f \in F\}$.

In a similar manner we define a (deterministic or nondeterministic) pushdown automaton with *standard mode*, *nondeterministic jumping mode* and *one-way*

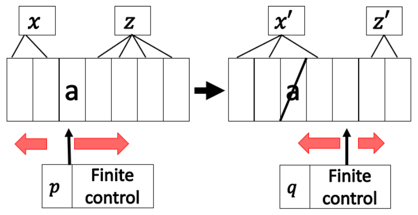

Fig. 1. Nondeterministic jumping mode of tape head move.

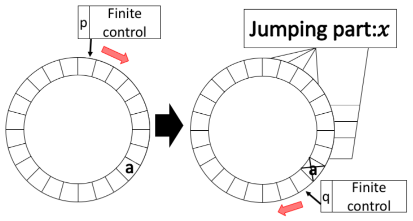

Fig. 2. One-way jumping mode of tape head move.

jumping mode, respectively, to be the rewriting systems $(M, \rightarrow)$, $(M, \curvearrowright)$ and $(M, \circlearrowright)$, respectively, where M is a (deterministic or nondeterministic) pushdown automaton.

2.2 Language Classes

We consider deterministic and nondeterministic finite automata and deterministic and nondeterministic pushdown automata, denoted by DFA, NFA, DPDA, NPDA, respectively. Then we classify these automata with three modes of tape head from the standpoint of languages accepted. We denote the language classes accepted by DFA, NFA, DPDA, NPDA with three modes $\rightarrow, \curvearrowright, \circlearrowright$ by $(\rightarrow, \curvearrowright, \circlearrowright)$-**DFA**, $(\rightarrow, \curvearrowright, \circlearrowright)$-**NFA**, $(\rightarrow, \curvearrowright, \circlearrowright)$-**DPDA**, $(\rightarrow, \curvearrowright, \circlearrowright)$-**NPDA**, respectively, in this paper. For example, $\rightarrow$**DFA** coincides with $\rightarrow$**NFA** and they comprise the class of regular languages, and $\rightarrow$**NPDA** is the class of context-free languages.

2.3 Differences of Modes of Tape Head Move

The next proposition shows the basic relationship between the processing of inputs by the same machine M in different tape head modes. Versions of the statement regarding the acceptance of inputs by M in the different modes have been shown in [13, Ch.17] and [4], for $\curvearrowright$ and $\circlearrowright$, respectively.

Proposition 1. *Let M be an (deterministic or non-deterministic) automaton. Suppose w_1, w_2, w_3 are words over Σ.*
(1) If $q_1w_1 \rightarrow^* q_2w_2$ then $q_1w_1 \curvearrowright^* q_2w_2$.
(2) If $q_1w_1 \rightarrow^* q_2w_2$ then $q_1w_1 \circlearrowright^* q_2w_2$.
(3) If $q_1w_1 \curvearrowright^* q_2w_2$ then there exists a permutation ϕ such that $q_1\phi(w_1) \rightarrow^* q_2w_2$.
(4) If $q_1w_1 \circlearrowright^* q_2w_2$ then there exists a permutation ϕ such that $q_1\phi(w_1) \rightarrow^* q_2w_2$.

Proof. (1) Let $q_1w_1 \rightarrow^n q_2w_2$ and let $m = |w_1|$. We prove that $q_1w_1 \curvearrowright^n q_2w_2$ holds, by induction on n. If $n = 0$, then $q_1 = q_2$ and $w_1 = w_2$. Thus, $q_1w_1 \curvearrowright^0 q_2w_2$ holds. Suppose that $n = k \leq m - 1$ holds. If n=k+1, then there exist q and $a \in \Sigma$ such that $q_1w_1 \rightarrow^k qaw_2 \rightarrow q_2w_2$. By $q_2 \in \delta(q, a)$, we have

$q_1w_1 \curvearrowright^k qaw_2 \curvearrowright q_2w_2 = q_1w_1 \curvearrowright^{k+1} q_2w_2$. Therefore, if $q_1w_1 \rightarrow^* q_2w_2$, then $q_1w_1 \curvearrowright^* q_2w_2$. We can prove (2) in a similar manner.

(3) Let $q_1w_1 \curvearrowright^n q_2w_2$ and let $m = |w_1|$. We prove that there exists a permutation ϕ such that $q_1\phi(w_1) \rightarrow^n q_2w_2$ holds, by induction on n. If $n = 0$, then there exists a permutation ϕ such that $\phi(w_1) = w_1$. Thus, $q_1\phi(w_1) \rightarrow^0 q_1w_1 = q_2w_2$ holds. Suppose that $n = k \leq m - 1$ holds. i.e. there exists $\sigma = \begin{pmatrix} 1 & 2 & 3 & \cdots & k-1 & k \\ \sigma(1) & \sigma(2) & \sigma(3) & \cdots & \sigma(k-1) & \sigma(k) \end{pmatrix}$ such that $q_1\sigma(w_1) \rightarrow^k q_2w_2$. If n=k+1, then there exist q and $a \in \Sigma$ such that $q_1w_1 \curvearrowright^k quav \curvearrowright q_2w_2$ where $uv = w_2$. From before, $q_1\sigma(w_1) \rightarrow^k quav \curvearrowright q_2w_2$ holds and let the permutation σ' be such that $\sigma'(xyaz) = xayz$ for $a \in \Sigma$ and $xyz \in \Sigma^*$ with $|x| = k$. Choosing $\phi = \sigma' \circ \sigma$ gives $q_1\phi(w_1) \rightarrow^k qauv \rightarrow q_2w_2$. Thus, $q_1\phi(w_1) \rightarrow^{k+1} q_2w_2$ holds.
We can prove (4) in a similar manner. □

Remark 1. It is easy to see that if $q_1w_1w_2 \rightarrow^* q_3$ and $q_1w_1 \rightarrow^* q_2$ then $q_2w_2 \rightarrow^* q_3$. However, $q_1w_1w_2 \curvearrowright^* q_3$ and $q_1w_1 \curvearrowright^* q_2$ does not imply $q_2w_2 \curvearrowright^* q_3$. Similarly, $q_1w_1w_2 \circlearrowright^* q_3$ and $q_1w_1 \circlearrowright^* q_2$ does not imply $q_2w_2 \circlearrowright^* q_3$.

In [1](equation (1)) the authors summarize the relationship between the languages accepted by DFA M in the various tape head modes, as:

$$L(M, \rightarrow) \subseteq L(M, \circlearrowright) \subseteq L(M, \curvearrowright) = Perm(L(M, \rightarrow)).$$

Regarding the $\rightarrow$ and $\curvearrowright$ modes, it makes no difference in the proofs whether the machine is deterministic. This means that we have the following relationship between acceptance in $\rightarrow$ and $\curvearrowright$ modes.

Proposition 2. *Let M be any automaton. A word w is accepted by $(M, \curvearrowright)$ if and only if there exists a permutation ϕ_w depending on w such that $\phi_w(w)$ is accepted by $(M, \rightarrow)$.*

We note that the permutation ϕ_w depends on the word w, however, it is not necessarily unique. Therefore, if a language L is accepted by a JFA $(M, \curvearrowright)$, we have $L = Perm(L)$.

Corollary 1. *A word w is accepted by a rewriting system $(M, \curvearrowright)$ if and only if there exists a permutation ϕ such that $\phi(w)$ is accepted by a rewriting system $(M, \rightarrow)$.*

The following corollary has been stated for JFA ($\curvearrowright$NFA) in [12] and for deterministic ROWJFA ($\circlearrowright$DFA) in [4]. They also follow, just as the nondeterministic case, from Proposition 1. A similar argument can easily be made for DPDA and NPDA.

Corollary 2. *Let M be a deterministic or nondeterministic finite automaton or pushdown automaton. Any word w, which is accepted by $(M, \rightarrow)$, is also accepted by $(M, \curvearrowright)$ and $(M, \circlearrowright)$.*

This means, generalizing the inclusions from before, that for any DFA, NFA, DPDA or NPDA M:

$$L(M, \rightarrow) \subseteq L(M, \circlearrowright) \subseteq L(M, \curvearrowright) = Perm(L(M, \rightarrow)). \tag{1}$$

Parikh's well-known paper [14] shows that all context-free languages have semilinear Parikh-image. Furthermore, all semilinear sets have a regular preimage, and this, combined with equation (1) tell us that the class of permutation closures of context-free languages is the same as the class of permutation closures of regular languages, accepted by DFA/NFA in $\curvearrowright$ mode ([5]). From here, by Corollary 1, we can deduce that in $\curvearrowright$ tape head mode, adding a stack to a finite automaton does not change the class of accepted languages, i.e.,

$$\curvearrowright\mathbf{DFA} = \curvearrowright\mathbf{DPDA} = \curvearrowright\mathbf{NPDA}.$$

Equation (1) also allows us to extend all the well-known pumping lemmas trivially to the language classes defined through the new execution modes.

Remark 2. Let M be an NFA, DFA, DPDA or NPDA, and $N_M > 0$ a constant which depends on M. If a pumping lemma holds for all words $w \in L(M, \rightarrow)$ with $|w| \geq N_M$, then for all words $w \in L(M, \curvearrowright) \cup L(M, \circlearrowright)$ with $|w| \geq N_M$, there exists a permutation ϕ such that the lemma holds for $\phi(w)$.

To close this section, we briefly discuss the special cases of unary and binary alphabets for the new tape head modes.

Theorem 1. *Let M be a deterministic or nondeterministic finite automaton or pushdown automaton with $|\Sigma| = 1$, then $L(M, \curvearrowright) = L(M, \circlearrowright) = L(M, \rightarrow)$.*

Proof. Straightforward, as the languages above are commutative and semilinear. □

It follows that there is no $\circlearrowright$NPDA that accepts $\{a^p \mid \text{p is a prime number}\}$, providing separation of $\circlearrowright$**NPDA** from the class of context-sensitive languages. Binary alphabets are also special cases for the $\curvearrowright$ mode. As shown in [10], over binary alphabets all commutative semilinear languages are context-free, so we have $\curvearrowright$**NFA**$\subset$$\rightarrow$**NPDA**. This is in contrast with larger alphabets, where $\curvearrowright$**NFA** and ($\rightarrow$)-**NPDA** are incomparable.

3 One-Way Jumping Nondeterministic Finite Automata ($\circlearrowright$NFA)

In previous papers [1, 2, 4] the class $\circlearrowright$**DFA** of languages accepted by one-way jumping deterministic finite automata has been investigated. In [1] it was shown that permutation closed languages in $\circlearrowright$**DFA** are characterized by having finitely many positive Myhill-Nerode equivalence classes. This nice characterization gave the corollary that a permutation closed language in $\circlearrowright$**DFA** is regular if and only if its complement is in $\circlearrowright$**DFA**. However, not much more has been known about the complements of $\circlearrowright$**DFA** languages. To tackle this, we look at the class

$\circlearrowright$**NFA** of languages accepted by one-way jumping nondeterministic finite automata ($\circlearrowright$NFA), that is, machines $(M, \circlearrowright)$, where M is an NFA without ϵ-moves, but with possibly multiple initial states. Then, we show that $\circlearrowright$**NFA** strictly includes $\curvearrowright$**NFA**, the class of permutation closed semilinear languages and, as a corollary, the complements of permutation closed $\circlearrowright$**DFA** languages. Finally, we go further by showing that $\circlearrowright$**NFA** contains the complements of all $\circlearrowright$**DFA** languages, but $\circlearrowright$**NFA** itself is not closed under complementation.

Definition 1. *($\circlearrowright$NFA) A (right) one-way jumping nondeterministic finite automaton is an ϵ-free NFA with multiple initial states in $\circlearrowright$ execution mode.*

Example 1. The language $K = \{w \in \{a, b\}^* : |w|_b = 0 \textit{ or } |w|_a = |w|_b\}$ is accepted by $\circlearrowright$NFA $M = (\{q_0, q_1, q_2, q_3, q_4, q_5\}, \{a, b\}, \delta, \{q_0\}, \{q_1, q_3\})$, where $\delta(q_0, a) = \{q_1, q_3\}, \delta(q_1, a) = \{q_2\}, \delta(q_2, b) = \{q_1\}, \delta(q_3, a) = \{q_3\}$.

In [1] it is shown that $K \notin \circlearrowright$**DFA**; this establishes $\circlearrowright$**DFA**$\subsetneq\circlearrowright$**NFA**. To move on to the inclusions in $\circlearrowright$**NFA** of the classes mentioned above, first we need to state that, as expected, $\circlearrowright$**NFA** is closed under union, by a construction similar to the case of classical NFA.

Proposition 3. *The class* $\circlearrowright$**NFA** *is closed under union.*

Proof. Let $M_1 = (Q_1, \Sigma_1, \delta_1, S_1, F_1)$ and $M_2 = (Q_2, \Sigma_2, \delta_2, S_2, F_2)$ be two NFA such that $Q_1 \cap Q_2 = \emptyset$ (if this does not hold, we can simply rename the states). It is straightforward to see that $M_3 = (Q_1 \cup Q_2, \Sigma_1 \cup \Sigma_2, \delta_1 \cup \delta_2, S_1 \cup S_2, F_1 \cup F_2)$ accepts the union of the languages accepted by M_1 and M_2 in $\circlearrowright$ execution mode, that is, $L(M_3, \circlearrowright) = L(M_1, \circlearrowright) \cup L(M_2, \circlearrowright)$. □

Theorem 2. *For any semilinear set $S \in \mathbb{N}^{|\Sigma|}$ there exists an NFA M, such that $\Psi_\Sigma^{-1}(S) = L(M, \circlearrowright)$, where Ψ_Σ is the Parikh mapping for alphabet Σ.*

Proof. S is semilinear, so we can write it as the finite union of linear sets. We give the construction for a linear set. The statement then follows from Proposition 3, by constructing disjoint NFA for all linear sets and taking their union to be M. Let $\Sigma = \{a_1, \ldots, a_k\}$ and $Lin \subseteq \mathbb{N}^k$ be a linear set of vectors over non-negative integers, i.e., there exist $v_0, \ldots, v_n \in \mathbb{N}^k$ such that

$$Lin = \{v_0 + c_1 v_1 + \cdots + c_n v_n \mid c_1, \ldots, c_n \in \mathbb{N}\}.$$

First we define the set of "starting vectors" as $Lin_0 = \{v_0, v_0 + v_1, \ldots, v_0 + v_n\}$, the minimal subset of Lin such that all letters which occur in Lin, also occur in Lin_0. The set $W(Lin_0)$ of representative words for a set of vectors Lin_0 will be a subset of the preimage of Lin_0 under the Parikh mapping. For each $(m_1, \ldots, m_k) \in Lin_0$ we take k words such that all possible starting letters are represented (there may be less than k if some $m_i = 0$). Several choices for $W : \mathbb{N}^{|\Sigma|} \to \Sigma^*$ are possible. We set W such that

$$W(Lin_0) = \bigcup_{(m_1,\ldots,m_k) \in Lin_0} \{a_1^{m_1} a_2^{m_2} \cdots a_k^{m_k}, a_2^{m_2} \cdots a_k^{m_k} a_1^{m_1}, \ldots, a_k^{m_k} a_1^{m_1} \cdots a_{k-1}^{m_{k-1}}\}$$

Now we construct M as follows. Let the initial state of M be s and for each $w_i \in W(Lin_0)$ add a new path from s to a new final state w_i labeled by letters of w_i (see Fig. 3). To each of these new final states w_i we add all possible loops labeled by words in $W(\{v_1, \ldots, v_n\})$ (see Fig. 3). Here we removed v_0, because it is only added once to each vector in Lin, according to the definition. It is clear

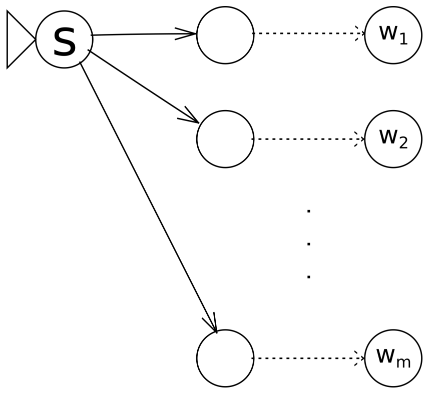

Fig. 3. For each $w_i \in Lin_0$, add a path to a final state, labeled by the letters of w_i.

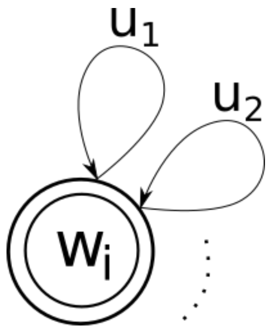

Fig. 4. For each $u_j \in W(\{v_1, \ldots, v_n\})$ add a loop to each w_i, labeled by the letters of u_j.

from the construction that for each vector $v \in Lin$ there is at least a path in M labeled by a word in the preimage of v. Conversely, the labels of each path form a word whose Parikh mapping is some $v \in Lin$. Even though not every preimage of v forms a path in M, the $\circlearrowright$ mode of execution allows M to read all the letters. □

At first sight it may seem that adding multiple paths for each vector in a linear set to the NFA in the previous proof is an overkill. While in some cases this could be avoided, as the next example shows, it is necessary in general.

Example 2. Consider the semilinear set, which is the union of linear sets $Lin_1 = \{(1,0,0,0) + c_1 \cdot (0,2,2,0) + c_2 \cdot (0,0,2,2) + c_3 \cdot (0,2,0,2) \mid c_1, c_2, c_3 \in \mathbb{N}\}$ and $Lin_2 = \{(0,1,0,0) + c_1 \cdot (2,0,0,0) + c_2 \cdot (2,2,0,0) \mid c_1, c_2 \in \mathbb{N}\}$ over alphabet $\{a,b,c,d\}$. If in the first phase of the construction only v_0 was to be added as a path from s for each linear set, then for input *bbcca*, the machine would have to choose the branch of Lin_2, because the first letter to the right for which there is a transition from s is b. This would mean *bbcca* is rejected, because $(1,2,2,0) \notin Lin_2$, even though $(1,2,2,0) \in Lin_1$ and thus $bbcca \in \Psi^{-1}(Lin_1 \cup Lin_2)$. If in the second phase of the construction only one path per vector v_i were to be added to the states w_i, then one of the inputs *abbcc*, *accdd* or *addbb* would be rejected even though they are in the language.

Corollary 3. $\curvearrowright$**NFA**$\subset\circlearrowright$**NFA**.

Proof. As it was shown in [4], $\curvearrowright$**NFA** is the class of permutation closed semilinear languages, that is, the class of languages which are preimages of semilinear sets under the Parikh mapping. □

Theorem 3. *For any DFA $M = (Q, \Sigma, \delta, s, F)$, we can construct an NFA M' such that $\Sigma^* \setminus L(M, \circlearrowright) = L(M', \circlearrowright)$.*

Proof. As we mentioned before, Σ_q denotes the set of letters for which there is an outgoing transition from $q \in Q$. We construct the NFA $M' = (Q', \Sigma, \delta', \{s, s'\}, F')$ by the following steps:

1. switch the accepting states and non-accepting states of M;
2. $\forall q \in Q \setminus F$: if $\Sigma_q \neq \Sigma$, then add new state $q' \in F'$, and the transitions:
   - $\delta'(p, a) = q'$ for all $p \in Q, a \in \Sigma$ such that $q \in \delta(p, a)$
   - $\delta'(q', b) = q'$, for each $b \in \Sigma \setminus \Sigma_q$;
3. $\forall q \in F$: if $\Sigma_q \neq \Sigma$, then add new states $q' \in F'$ and $q'' \notin F'$, and the transitions:
   - $\delta'(p, a) = q''$ for all $p \in Q, a \in \Sigma$ such that $q \in \delta(p, a)$
   - $\delta'(q'', b) = q'$ and $\delta'(q', b) = q'$, for each $b \in \Sigma \setminus \Sigma_q$;
4. if $\Sigma_s \neq \Sigma$ then if $s \in F$, let s'' be a new initial state, whereas if $s \notin F$, let s' be a new initial state.

For each $w \in \Sigma^*$, the machine M' will read all of the input along some nondeterministically chosen path. Some path will finish in an accepting state $q \in F' \cap Q$ if and only if $(M, \circlearrowright)$ read the whole input and stopped in a non-accepting state. Some path will finish in an accepting state $q' \in F' \setminus Q$ if and only if $(M, \circlearrowright)$ could not read the input and got stuck in state q, because the remaining letters were all elements of Σ_q. In more detail, for each input w one of the following happens:

- $w \in L(M, \circlearrowright)$: this means M reads all the input and finishes in a state from $q \in F$. The only branches of M' which read the whole input finish in q or q'' (if it exists), but since $q, q'' \notin F'$, the input is rejected by M';
- $w \notin L(M, \circlearrowright)$ and M reads the whole input finishing in some state $q \notin F$. In this case, the same path in M' reads the whole input and finishes in $q \in F'$, so M' accepts the input;
- $w \notin L(M, \circlearrowright)$ and M cannot read the whole input, so M gets stuck in a state q, with the remaining input being in $(\Sigma \setminus \Sigma_q)^*$. Depending on whether q is final or not, we distinguish two subcases:
  - $q \in F$: from the states preceding q in the path, the branch that goes to q gets stuck. The branch that goes to q'' reads the remaining input after transitioning to $q' \in F'$, so M' accepts;
  - $q \notin F$: from the states preceding q in the path, the branch that goes to q gets stuck. The branch that goes to $q' \in F'$ reads the remaining input, so M' accepts.

□

Example 3. To illustrate how the construction works (Fig. 5), consider the non-regular $\circlearrowright$**DFA** language $L_{ab} = \{w \in \{a, b\}^* \mid |w|_a = |w|_b\}$, which can be accepted by the $\circlearrowright$DFA $M = (\{s, q\}, \{a, b\}, \delta, s, \{s\})$, where $\delta(s, a) = q$ and $\delta(q, b) = s$.

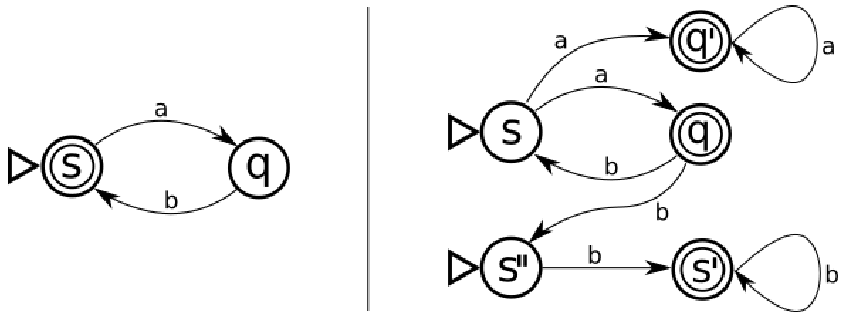

Fig. 5. Left: ⟳DFA for $L_{ab} = \{w \in \{a,b\}^* \mid |w|_a = |w|_b\}$. Right: ⟳NFA for $\Sigma^* \setminus L_{ab}$.

4 One-Way Jumping Pushdown Automaton: (⟳)-NPDA, (⟳)-DPDA

In this section we initiate the study of how the ⟳ tape head mode affects the computational power of PDA. In particular, we exhibit certain languages which provide separation between the classes of languages accepted by DPDA and NPDA in the ⟳ mode and their classical counterparts, as well as, finite automata in ⟳ mode. We also present languages which show that several common closure properties do not apply for **⟳NPDA**. The proofs for the results that follow are based on extended versions of pumping lemmas for deterministic context-free languages ([16]) and context-free languages (Bar-Hillel).

First, we state two extended pumping lemmas for ⟳ mode. The proofs of these lemmas are trivial based on Remark 2.

Corollary 4 (Bar-Hillel lemma for ⟳-NPDA). *For any language L accepted by a ⟳NPDA there exists a constant n, such that for every string $w \in L$ with $|w| > n$, there exists a permutation w_σ, which can be written as $w_\sigma = uvxyz$, satisfying (1) $|vy| \geq 1$, (2) $|vxy| \leq n$ and (3) $uv^ixy^iz \in L$ for every $i \geq 0$.*

Corollary 5 (Pumping Lemma for ⟳DPDA, original version in [16]). *Suppose L is accepted by a ⟳DPDA M. Then there exists a constant n for L such that for any pair of words $w, w' \in L$ if*

(1) $s = xy$ and $s' = xz$, $|x| > n$, and
(2) (first symbol of y) = (first symbol of z),

where s and s' are permutations of w and w', such that $s, s' \in L(M, \rightarrow)$, then either (3) or (4) holds:

(3) there is a factorization $x = x_1x_2x_3x_4x_5$, $|x_2x_4| \geq 1$ and $|x_2x_3x_4| \leq n$, such that for all $i \geq 0, x_1{x_2}^ix_3{x_4}^ix_5y$ and $x_1{x_2}^ix_3{x_4}^ix_5z$ are in L;
(4) there exist factorizations $x = x_1x_2x_3, y = y_1y_2y_3$ and $z = z_1z_2z_3, |x_2| \geq 1$ and $|x_2x_3| \leq n$, such that for all $i \geq 0, x_1{x_2}^ix_3y_1{y_2}^iy_3$ and $x_1{x_2}^ix_3z_1{z_2}^iz_3$ are in L.

Let $L_{\text{ppal}} = \{w\#\phi(w) \mid w \in \{a,b\}^*, \phi \in S_{|w|}\}$. Note that $L_{\text{ppal}} \notin \rightarrow\mathbf{NPDA}$, by a simple application of the Bar-Hillel lemma. By a construction mimicking the DPDA accepting palindromes $w\#w^R$, it is easy to show that $L_{\text{ppal}} \in \circlearrowleft\mathbf{DPDA}$.

The following propositions provide us with the separation results which, added to the previously known relationships, add up to Fig. 6.

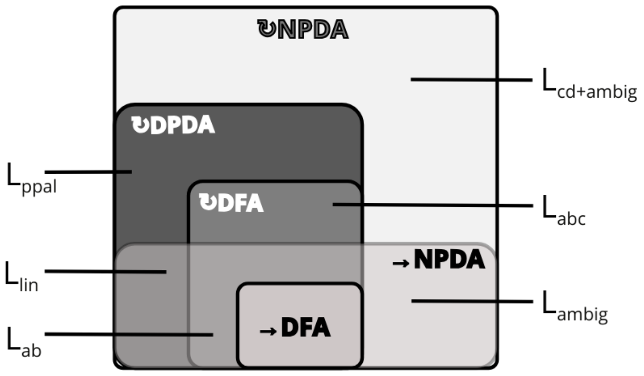

Fig. 6. Relationship between REG=→DFA=→NFA, CF=→NPDA, and one-way jumping classes.

Proposition 4. $L_{\text{ppal}} \notin \circlearrowleft\mathbf{DFA}$.

Proof. Suppose that $M = (Q, \Sigma, \delta, q_0, F)$ is a DFA, such that $L(M, \circlearrowleft) = L_{\text{ppal}}$. Consider the word $w = a^p\#a^p \in L_{\text{ppal}}$, with $p = |Q| + 1$. By Proposition 1 there exists a permutation P such that $P(w) \in L(M, \rightarrow) \subseteq L_{\text{ppal}}$ and the pumping lemma for regular languages says that it can be written as $P(w) = xyz$, where $y \neq \epsilon, |xy| \leq |Q|$ and $xy^iz \in L_{\text{ppal}}, \forall i \geq 0$. Since $L_{\text{ppal}} \cap (a + \#)^*$ has only one word of each length, we have $P(w) = a^p\#a^p$. From $|xy| \leq |Q|$, we get a contradiction when $i = 2$ as $xy^iz \notin L_{\text{ppal}}$.

□

Proposition 5. $L_{\text{ambig}} = \{a^ib^i | i \geq 0\} \cup \{a^ib^{2i} | i \geq 0\} \notin \circlearrowleft\mathbf{DPDA}$.

Proof. Assume there exists DPDA M=$(Q, \Sigma, \Gamma, \delta, q_0, F)$ such that $L(M, \circlearrowleft) = L_{\text{ambig}}$ and let C be the constant for L_{ambig} in Corollary 5.

Choose $w = a^nb^n$ and $w' = a^nb^{2n}$ for some integer $n > C$, then there exist permutations σ and σ' such that $w_\sigma, w'_{\sigma'} \in L(M, \rightarrow)$. By $w_\sigma, w'_{\sigma'} \in L(M, \rightarrow) \subseteq L(M, \circlearrowleft) = L_{\text{ambig}}$, we get that $w_\sigma = a^nb^n$ and $w'_{\sigma'} = a^nb^{2n}$. Let $x = a^nb^{n-1}$, $y = b$, and $z = b^{2n-1}$. The choice of $w_\sigma = xy$ and $w'_{\sigma'} = xz$ satisfies (1) and (2) of Corollary 5. According to Corollary 5, either (3) or (4) should hold.

Let us consider (3) first. The only possible factorization $x = x_1x_2x_3x_4x_5$ such that $|x_2x_4| > 0$ and for all i, $x_1{x_2}^ix_3{x_4}^ix_5y \in L_{\text{ambig}}$ must satisfy the condition

$x_2 = a^k$ and $x_4 = b^k$ for some $k > 0$. But then $x_1{x_2}^0x_3{x_4}^0x_5z = x_1x_3x_5z = a^{n-k}b^{2n-k} \notin L_{\text{ambig}}$. Therefore (3) does not hold.

Now,we consider (4). Any factorization $x = x_1x_2x_3$ such that $|x_2| > 0$ and $|x_2x_3| \leq C < n$ will result in $x_2 \in b^+$ and $y_2 \in b^*$, so $x_1x_3y_1y_3 = a^nb^{n-|x_2|-|y_2|} \notin L_{\text{ambig}}$. So (4) does not hold either.

This contradicts the $\circlearrowleft$-DPDA pumping lemma, so $L_{\text{ambig}} \notin \circlearrowleft\mathbf{DPDA}$. $\square$

The language L_{ambig} is a classic example of nondeterministic context-free language. At the same time, as mentioned in [4],

$$L_{abc} = \{w \in \{a,b,c\}^* \mid |w|_a = |w|_b = |w|_c\} \in \circlearrowleft\mathbf{DFA} \setminus \rightarrow\mathbf{NPDA}.$$

It is also straightforward that

$$L_{\text{lin}} = \{a^nb^n \mid n > 0\} \in \circlearrowleft\mathbf{DPDA} \cap \rightarrow\mathbf{NPDA},$$

and $L_{\text{lin}} \notin \circlearrowleft\mathbf{DFA}$, by [4, Cor.12]. Since $\rightarrow\mathbf{NPDA}$ is trivially included in $\circlearrowleft\mathbf{NPDA}$, by the same arguments as above, the union $L_{cd} \cup L_{\text{ambig}} = L_{cd+\text{ambig}}$, where $L_{cd} = \{w \in \{c,d\}^* \mid |w|_c = |w|_d\}$, is in $\circlearrowleft\mathbf{NPDA}$. At the same time, it is easy to see that $L_{cd+\text{ambig}} \notin (\rightarrow\mathbf{NPDA} \cup \circlearrowleft\mathbf{DPDA})$. This completes the separation examples for Fig. 6. Finally, let us say a few words about closure properties of $\circlearrowleft\mathbf{NPDA}$.

Proposition 6. $L_1 = \{a^nb^nc^n | n \geq 0\} \notin \circlearrowleft\mathbf{NPDA}$.

Proof. Suppose that M$=(Q, \{a,b,c\}, \Gamma, \delta, q_0, F)$ and $L(M, \circlearrowleft) = L_3$. Consider the word $w = a^nb^nc^n \in L_1$. By Corollary 4 there exists a permutation σ such that $w_\sigma \in L_1$ can be written as $w_\sigma = uvxyz$, where $|vy| \geq 1$, $|vxy| \leq n$, $uv^ixy^iz \in L_1$, for all $i \geq 0$, where n is the contant from the Bar-Hillel lemma for $\circlearrowleft\mathbf{NPDA}$. Depending on the decomposition $uvxyz$, we have two cases

1. If vxy is generated by one symbol, then uv^2xy^2z does not include the same number of a, b, c. This contradicts $uv^ixy^iz \in L_1$, for $i = 2$.
2. If vxy contains two kinds of symbols, then $uv^2xy^2z \notin L_1$, because the number of copies of the third letter (the one not in vxy) does not match the other two. This contradicts $uv^ixy^iz \in L_1$, when $i = 2$.

Therefore, no $\circlearrowleft$NPDA accepts L_1. $\square$

Since both $L_{abc} = \{w \mid |w|_a = |w|_b = |w|_c\} \in \circlearrowleft\mathbf{NPDA}$ and $a^*b^*c^* \in \circlearrowleft\mathbf{NPDA}$, from Proposition 6 we get that $\circlearrowleft\mathbf{NPDA}$ is not closed under intersection. Together with the fact that it is closed under union, we get that it is not closed under complementation, either.

Theorem 4. $L_2 = \{wa \mid |w|_a = |w|_b = |w|_c\} \notin \circlearrowleft\mathbf{NPDA}$.

Proof. Suppose that there exist a $\circlearrowleft$NPDA $M = (Q, \{a,b,c\}, \Gamma, \delta, q_0, F)$, which accepts L_2. Consider the words $w_1 = a^mb^mc^ma \in L_2$ and $w_2 = a^{m+1}b^mc^m \notin L_2$, where m is the constant from Corollary 4. By $w_1 = a^mb^mc^ma \in L_2$, there exists permutation P such that $P(w_1) \in L(M, \rightarrow)$. Let $P(w_1) = x_1x_2...x_{3m+1}$, then $x_{3m+1} = a$ by $P(w_1) \in L_2$. We will use the fact that w_2 is the cyclic shift of

w_1. Let us look at the computation performed by the automaton on reading w_1. Before reading b or c, the automaton will read a^k for some $k \geq 0$, that is:

$$(q_0, a^m b^m c^m a, \$) \circlearrowright (q_1, a^{m-1} b^m c^m a, z_1) \circlearrowright .... \circlearrowright (q_k, a^{m-k} b^m c^m a, z_k)$$

for some $\{q_1, q_2 ... q_k\} \subset Q$ and $\{z_1, z_2 .... z_k\} \subset \Gamma^*$ (\$ is the bottom marker for the pushdown). Since $w \in L$, there is an accepting computation from the last configuration $(q_k, a^{m-k} b^m c^m a, z_k)$ in that sequence. Depending on k, we have two cases.

(CASE 1) if $k < m$: we get that there is no transition defined from state q_k on reading a, therefore $a^{m-k}, a^{m-k+1} \in (\Sigma - \Sigma_{(q_k, z_k)})^*$, where $\Sigma_{(q_k, z_k)} = \{d \in \Sigma \mid \delta(q_k, d, z_k) \neq \emptyset\}$. We get that

$$(q_k, uv, z_k) = (q_k, a^{m-k+1} b^m c^m, z_k) \circlearrowright^* (q, \epsilon, z)$$

if and only if

$$(q_k, vu, z_k) = (q_k, b^m c^m a^{m-K+1}, z_k) \circlearrowright^* (q, \epsilon, z).$$

This means $(q_0, a^{m+1} b^m c^m, \$) \circlearrowright^* (q_k, a^{m-k+1} b^m c^m, z_k) \circlearrowright^* (q, \epsilon, z)$, which contradicts the initial assumption $a^{m+1} b^m c^m \notin L_2$.

(CASE 2) if $k = m$: This case says that $(q_0, a^m b^m c^m a, \$) \circlearrowright^k (q_m, b^m c^m a, z_m)$.

Let $\circlearrowright$NPDA M then read b. Let us look at the computation performed by the automaton on reading $b^m c^m a$. Before reading c, the automaton will read b^l for some $l \geq 0$, that is:

$$(q_m, b^m c^m a, z_m) \circlearrowright (q_{m+1}, b^{m-1} c^m a, z_{m+1}) \circlearrowright .... \circlearrowright (q_{m+l}, b^{m-l} c^m a, z_{m+l})$$

for some $\{q_{m+1}, q_{m+2} ... q_{m+l}\} \subset Q$ and $\{z_{m+1}, z_{m+2} .... z_{m+k}\} \subset \Gamma^*$. There is an accepting computation from the last configuration $(q_{m+l}, b^{m-l} c^m a, z_{m+l})$ in that sequence. Depending on l, we have two cases.
(CASE 2-1) if $l < m$: This case is the same as (CASE 1).
(CASE 2-2) if $l = m$: This case is the same as (CASE 2). $\circlearrowright$NPDA M must read c. Let us look at the computation performed by the automaton on reading $c^m a$. Before reading the last letter, a, the automaton will read c^n for some $n \geq 0$:

$$(q_{2m}, c^m a, z_{2m}) \circlearrowright^n (q_{2m+n}, c^{m-n} a, z_{2m+n})$$

for some $q_{2m+n} \in Q$ and $z_{2m+n} \in \Gamma^*$. There is an accepting computation from the last configuration $(q_{2m+n}, c^{m-n} a, z_{2m+n})$ in that sequence. Depending on n, we have two cases.
(CASE 2-2-1) if $n < m$: This case is the same as (CASE 1).
(CASE 2-2-2) if $n = m$: This case says that

$$(q_{2m}, c^m a, z_{2m}) \circlearrowright^m (q_{3m}, a, z_{3m}) \circlearrowright (q_{3m+1}, \epsilon, z_{3m+1})$$

for $\{q_{3m}, q_{3m+1}\} \subseteq Q$ and $\{z_{3m}, z_{3m+1}\} \subseteq \Gamma^*$. In this case, $P(w_1) = a^m b^m c^m a \in L(M, \rightarrow)$, so we can write $a^m b^m c^m a = uvxyz$, where $|vy| \geq 1$, $|vxy| \leq m$, $uv^i xy^i z \in L_2$, for all $i \geq 0$. The same argument as in the proof of Proposition 6 can be applied to reach a contradiction.

Therefore, no $\circlearrowright$NPDA accepts L_2.

□

Since $L_{abc} \in$$\circlearrowright$**NPDA** and $\{a\} \in$$\circlearrowright$**NPDA**, from Theorem 4 we get that the class $\circlearrowright$**NPDA** is not closed under concatenation. However, the reversal of L_2, that is, $\{aw \mid |w|_a = |w|_b = |w|_c\}$, is in $\circlearrowright$**NPDA**, in fact, it is in $\circlearrowright$**DFA**, so we get that $\circlearrowright$**NPDA** is not closed under reversal, either.

5 Summary

We discussed three modes of tape head, however, other established modes can be considered, e.g., a two-way jumping mode. In such a mode, the tape head does not erase the letters it read. The modes can be applied to linear bounded automata, but one can show that none of these modes, including the two-way jumping mode adds to the power of linear bounded automata. Therefore, the language accepted by linear bounded automata with the mentioned alternative modes of tape head coincides with the class of context sensitive languages. Several questions remain open with respect to the jumping modes, particularly so in the case of $\circlearrowright$. An unanswered decidability problem, which so far resisted attempts, is whether there exists an algorithm which decides $L(M, \circlearrowright) \in$REG for a given DFA/NFA M. Here, the technique of completing $\circlearrowright$DFA presented in Section 3 might help, but the problem seems rather difficult, because of the unusual languages which these classes of machines can accept.

References

1. Beier, S., Holzer, M.: Decidability of right one-way jumping finite automata. In: Hoshi, M., Seki, S. (eds.) Developments in Language Theory - 22nd International Conference, DLT 2018, Tokyo, Japan, September 10-14, 2018, Proceedings. Lecture Notes in Computer Science, vol. 11088, pp. 109–120. Springer (2018), `https://doi.org/10.1007/978-3-319-98654-8\_9`
2. Beier, S., Holzer, M.: Properties of right one-way jumping finite automata. In: Konstantinidis, S., Pighizzini, G. (eds.) Descriptional Complexity of Formal Systems - 20th IFIP WG 1.02 International Conference, DCFS 2018, Halifax, NS, Canada, July 25-27, 2018, Proceedings. Lecture Notes in Computer Science, vol. 10952, pp. 11–23. Springer (2018), `https://doi.org/10.1007/978-3-319-94631-3\_2`
3. Beier, S., Holzer, M., Kutrib, M.: Operational state complexity and decidability of jumping finite automata. In: Charlier, É., Leroy, J., Rigo, M. (eds.) Developments in Language Theory - 21st International Conference, DLT 2017, Liège, Belgium, August 7-11, 2017, Proceedings. Lecture Notes in Computer Science, vol. 10396, pp. 96–108. Springer (2017), `https://doi.org/10.1007/978-3-319-62809-7\_6`

4. Chigahara, H., Fazekas, S.Z., Yamamura, A.: One-way jumping finite automata. Int. J. Found. Comput. Sci. **27**(3), 391–405 (2016), `https://doi.org/10.1142/S0129054116400165`
5. Fernau, H., Paramasivan, M., Schmid, M.L.: Jumping finite automata: Characterizations and complexity. In: Drewes, F. (ed.) Implementation and Application of Automata - 20th International Conference, CIAA 2015, Umeå, Sweden, August 18-21, 2015, Proceedings. Lecture Notes in Computer Science, vol. 9223, pp. 89–101. Springer (2015), `https://doi.org/10.1007/978-3-319-22360-5\_8`
6. Fernau, H., Paramasivan, M., Schmid, M.L., Vorel, V.: Characterization and complexity results on jumping finite automata. Theor. Comput. Sci. **679**, 31–52 (2017), `https://doi.org/10.1016/j.tcs.2016.07.006`
7. Hopcroft, J.E., Ullman, J.D.: Introduction to Automata Theory, Languages and Computation. Addison-Wesley (1979)
8. Kocman, R., Meduna, A.: On parallel versions of jumping finite automata. In: Advances in Intelligent Systems and Computing, pp. 142–149. Springer International Publishing (2016), `https://doi.org/10.1007/978-3-319-46535-7\_12`
9. Krivka, Z., Meduna, A.: Jumping grammars. Int. J. Found. Comput. Sci. **26**(6), 709–732 (2015), `https://doi.org/10.1142/S0129054115500409`
10. Latteux, M.: Cônes rationnels commutatifs. J. Comput. Syst. Sci. **18**(3), 307–333 (1979), `https://doi.org/10.1016/0022-0000(79)90039-4`
11. Madejski, G.: Jumping and pumping lemmas and their applications. In: Eighth Workshop on Non-Classical Models of Automata and Applications (NCMA 2016) Short papers. pp. 25–33 (2016)
12. Meduna, A., Zemek, P.: Jumping finite automata. Int. J. Found. Comput. Sci. **23**(7), 1555–1578 (2012), `https://doi.org/10.1142/S0129054112500244`
13. Meduna, A., Zemek, P.: Regulated Grammars and Automata. Springer (2014), `https://doi.org/10.1007/978-1-4939-0369-6`
14. Parikh, R.: On context-free languages. J. ACM **13**(4), 570–581 (1966), `https://doi.org/10.1145/321356.321364`
15. Sipser, M.: Introduction to the Theory of Computation. Course Technology, second edn. (2006)
16. Yu, S.: A pumping lemma for deterministic context-free languages. Inf. Process. Lett. **31**(1), 47–51 (1989), `https://doi.org/10.1016/0020-0190(89)90108-7`