Nondeterministic Right One-Way Jumping Finite Automata (Extended Abstract)

Simon Beier, Markus Holzer

Nondeterministic Right One-Way Jumping Finite Automata (Extended Abstract)

Simon Beier and Markus Holzer

Institut für Informatik, Universität Giessen,
Arndtstr. 2, 35392 Giessen, Germany
email: {simon.beier,holzer}@informatik.uni-giessen.de

Abstract. Right one-way jumping finite automata are deterministic devices that process their input in a discontinuous fashion. We generalise these devices to nondeterministic machines. More precisely we study the impact on the computational power of these machines when allowing multiple initial states and/or a nondeterministic transition function including spontaneous or λ-transitions. We show inclusion relations and incomparability results of the induced language families. Since for right-one way jumping devices the use of spontaneous transitions is subject to different natural interpretations, we also study this subject in detail, showing that most interpretations are equivalent to each other and lead to the same language families. Finally we also study inclusion and incomparability results to classical language families and to the families of languages accepted by finite automata with translucent letters.

1 Introduction

Right one-way jumping finite automata (ROWJFAs) were introduced in [4] as a deterministic variant of jumping finite automata [9], a machine model for discontinuous information processing, that is allowed to read letters from anywhere in the input string, not necessarily only from left of the remaining input. In a right one-way jumping finite automaton the device moves the input head deterministically from left-to-right starting from the leftmost letter in the input and when it reaches the end of the input word, it returns to the beginning and continues the computation. The language families induced by these two automata models are quite interesting since they have relations to semi-linear sets, see, e.g., [1–3, 5, 6, 11]. While languages that are accepted by jumping finite automata are permutation closed and semi-linear [9], the permutation closed languages that are accepted by ROWJFAs are exactly those languages with a finite number of positive Myhill-Nerode equivalence classes [1], that is, the permutation closed language under consideration can be written as the finite union of Myhill-Nerode classes. This is a nice transfer of a classical result on finite automata that states that a language is regular or accepted by a deterministic finite automaton if and only if the number of Myhill-Nerode equivalence classes is finite, to the case where the index (number of equivalence classes) of the Myhill-Nerode relation is not finite any more. Since many generalizations of finite automata such as,

e.g., nondeterminism, multiple initial states, etc. do note change their accepting power, the question arises whether a similar result is valid for ROWJFAs, too? We answer this question for two types of generalizations, namely nondeterministic transitions and/or multiple initial states, for jumping finite automata and ROWJFAs.

For jumping finite automata the generalizations to nondeterministic transitions functions and/or multiple initial states do not change the computational power of these devices and leads to the language family **JFA**. This is in sharp contrast to ROWJFAs, where these generalizations increase the computational power of these devices and thus induce a mesh of language families and their permutation closed variants that are strictly included within each other (due to the trivial inclusion relations), subject to one exception. For the case of nondeterministic ROWJFAs and nondeterministic ROWJFAs with multiple initial states, we find that the induced languages families **NROWJ** and **MNROWJ** coincide in case we consider permutation closed languages. Then this language family is nothing other than **JFA**, the family of all languages accepted by jumping finite automata. The corresponding deterministic language families **ROWJ** and **MROWJ** of all languages accepted by ROWJFAs and those with multiple initial states satisfy $\mathbf{ROWJ} \subset \mathbf{MROWJ}$ and $\mathbf{pROWJ} \subset \mathbf{pMROWJ}$, resp., where the prefix **p** refers to the permutation closed language family in question. By results in [2], it was shown that the permutation closed languages over binary alphabets accepted by ROWJFAs with multiple initial states are those languages that are a finite union of permutation closed languages, where each language has a finite number of positive Myhill-Nerode equivalence classes. For arbitrary alphabets we have a characterization which is a bit more sophisticated and uses the quotient of a language by a word. Although the generalization of ROWJFAs to nondeterministic ROWJFAs with and without multiple initial states is straightforward, it is not clear how to describe the semantics of these machines in case of spontaneous or λ-transitions. This issue is discussed in detail and we give three different semantics, where it turns out that two of them are equivalent. The introduced generalizations nicely fit into the known landscape of the language families that were considered earlier. We further prove inclusion and incomparability results to classical language families such as the deterministic context-free languages, context-free languages, and context-sensitive languages. In some of our proofs a nice and tight relation between languages accepted by ROWJFAs and variants thereof to semi-linear sets and languages with finite positive Myhill-Nerode equivalence classes [1, 2] is exploited.

Because ROWJFAs share some features of finite automata with translucent letters [10] since their input letter reading mechanism looks similar, it is natural to consider the relation of the language families given by these automata to ROWJFAs and generalizations thereof. In passing we solve an open problem on deterministic finite automata with translucent letters stated in [10]. Due to space constraints all proofs are omitted and will be given in a journal version of this paper.

2 Preliminaries

We assume the reader to be familiar with the basics in automata and formal language theory as contained, for example, in [7]. Let $\mathbb{N} = \{0, 1, 2, \ldots\}$ be the set of non-negative integers. We use $\subseteq$ for inclusion and $\subset$ for proper inclusion of sets. We denote the powerset of a set S by 2^S. For a binary relation $\sim$ let $\sim^+$ and $\sim^*$ denote the transitive closure of $\sim$ and the transitive-reflexive closure of $\sim$, respectively. In the standard manner, $\sim$ is extended to $\sim^n$, where $n \geq 0$. Let Σ be an alphabet. Then Σ^* is the set of all words over Σ, including the empty word λ. For a language $L \subseteq \Sigma^*$ define the set $\mathsf{perm}(L) = \cup_{w \in L} \mathsf{perm}(w)$, where $\mathsf{perm}(w) = \{ v \in \Sigma^* \mid v \text{ is a permutation of } w \}$. A language L is called *permutation closed* if $L = \mathsf{perm}(L)$. The length of a word $w \in \Sigma^*$ is denoted by $|w|$. For the number of occurrences of a symbol a in w we use the notation $|w|_a$. If Σ is the ordered alphabet $\Sigma = \{a_1, a_2, \ldots, a_k\}$, the function $\psi : \Sigma^* \to \mathbb{N}^k$ with $w \mapsto (|w|_{a_1}, |w|_{a_2}, \ldots, |w|_{a_k})$ is called the *Parikh-mapping*. The set $\psi(L)$ is called the *Parikh-image* of L. Two languages over the same ordered alphabet are said to be *Parikh-equivalent* if and only if they have the same Parikh-image. The language L is called *letter-bounded* if and only if $L \subseteq a_1^* a_2^* \cdots a_k^*$.

For a vector $\boldsymbol{c} \in \mathbb{N}^k$ and a finite set $P \subset \mathbb{N}^k$ let $L(\boldsymbol{c}, P)$ denote the subset $L(\boldsymbol{c}, P) = \{ \boldsymbol{c} + \sum_{\boldsymbol{x_i} \in P} \lambda_i \cdot \boldsymbol{x_i} \mid \lambda_i \in \mathbb{N} \}$ of $\mathbb{N}^k$. Sets of the form $L(\boldsymbol{c}, P)$, for a $\boldsymbol{c} \in \mathbb{N}^k$ and a finite $P \subset \mathbb{N}^k$, are called *linear* subsets of $\mathbb{N}^k$. A subset of $\mathbb{N}^k$ is said to be *semi-linear* if it is a finite union of linear subsets. Moreover, a language $L \subseteq \Sigma^*$ is called *semi-linear* if its Parikh-image $\psi(L)$ is semi-linear.

For an alphabet Σ and a language $L \subseteq \Sigma^*$, let $\sim_L$ be the *Myhill-Nerode equivalence relation* on Σ^*. So, for $v, w \in \Sigma^*$, we have $v \sim_L w$ if and only if, for all $u \in \Sigma^*$, the equivalence $vu \in L \Leftrightarrow wu \in L$ holds. For $w \in \Sigma^*$, we call the equivalence class $[w]_{\sim_L}$ *positive* if and only if $w \in L$.

We define a *nondeterministic finite automaton with multiple start states and spontaneous or λ-transitions*, a λ-MNFA for short, as a tuple $A = (Q, \Sigma, R, S, F)$, where Q is the finite *set of states*, Σ is the finite *input alphabet*, R is a function from $Q \times (\Sigma \cup \{\lambda\})$ to 2^Q, $S \subseteq Q$ is the set of *start states*, and $F \subseteq Q$ is the set of *final states*. The elements of R are referred to as *rules* of A and we simply write $pa \to q \in R$ instead of $q \in R(p, a)$, for $p, q \in Q$ and $a \in \Sigma \cup \{\lambda\}$.

A *configuration* of A is a string in $Q\Sigma^*$. The automaton A makes a *transition* from configuration paw to configuration qw if $pa \to q \in R$, where $p, q \in Q$, $a \in \Sigma \cup \{\lambda\}$, and $w \in \Sigma^*$. We denote this by $paw \vdash_A qw$ or just $paw \vdash qw$ if it is clear which automaton we are referring to. The *language accepted by* A is

$$L(A) = \{ w \in \Sigma^* \mid \exists s \in S, f \in F : sw \vdash^* f \}.$$

We say that A *accepts* $w \in \Sigma^*$ if $w \in L(A)$ and that A *rejects* w otherwise.

The automaton A is called a *nondeterministic finite automaton with multiple start states*, an MNFA, if $R(p, \lambda) = \emptyset$ for all $p \in Q$. We call A a *nondeterministic finite automaton with λ-transitions*, a λ-NFA, if $|S| = 1$. If A is an MNFA and a λ-NFA, we say that A is a *nondeterministic finite automaton*, an NFA. The automaton A is said to be a *deterministic finite automaton with multiple start*

states, an MDFA, if A is an MNFA and $|R(p,a)| \leq 1$ for all $p \in Q$ and $a \in \Sigma$. Finally, we call A a *deterministic finite automaton*, a DFA, if it is an MDFA and an NFA. It is well known that for all the aforementioned types of automata are computational equivalent and the family of languages accepted by any of these devices is referred to as the family of regular languages **REG**. Observe that these automata read the input in a symbol by symbol manner from left to right.

Besides the family of regular languages **REG**, we also use the following language families: let **DCF**, **CF**, and **CS** be the families of deterministic context-free, context-free, and context-sensitive languages. Moreover, we are interested in families of permutation closed languages. These language families are referred to by a prefix **p**. For instance, **pREG** denotes the language family of all permutation closed regular languages.

3 Variants of Nondeterministic Jumping Finite Automata

We start our investigation with the generalization of jumping automata to variants of jumping automata with multiple initial states. Originally, jumping finite automata were introduced in [9] as deterministic or nondeterministic devices with a sole initial state. The idea behind a jumping finite automaton is that the device no longer reads the input from left to right, but is allowed to read letters from anywhere in the input string. For a λ-MNFA $A = (Q, \Sigma, R, S, F)$ this behaviour can be modelled by a binary relation, the *jumping relation*, which is symbolically denoted by $\curvearrowright_A$, over *configurations* from $Q\Sigma^*$ and is defined as follows: consider an $a \in \Sigma \cup \{\lambda\}$, words $x, y \in \Sigma^*$, states $p, q \in Q$, and $pa \to q \in R$. Then, we write $pxay \curvearrowright_A qxy$ or just $pxay \curvearrowright qxy$ if it is clear which automaton we are referring to. The *language accepted by A interpreted as a jumping finite automaton* is

$$L_J(A) = \{\, w \in \Sigma^* \mid \exists s \in S, f \in F : sw \curvearrowright^* f \,\}.$$

We say that A interpreted as a jumping finite automaton *accepts* $w \in \Sigma^*$ if $w \in L_J(A)$, otherwise A interpreted as a jumping finite automaton *rejects* the word w.

Example 1. Let A be the DFA$A = (\{q_0, q_1, q_2, q_3\}, \{a, b\}, R, q_0, \{q_3\})$, where R consists of the rules $q_0 b \to q_1$, $q_0 a \to q_2$, $q_2 b \to q_3$, and $q_3 a \to q_2$. The automaton A is depicted in Figure 1. It holds

$$L(A) = (ab)^+ \quad \text{and} \quad L_J(A) = \mathsf{perm}\left((ab)^+\right) = \left\{\, w \in \{a,b\}^+ \;\middle|\; |w|_a = |w|_b \,\right\}.$$

Observe, that $L_J(A)$ is a permutation closed language that is non-regular but context-free. □

It can easily be seen that $L_J(A) = \mathsf{perm}(L(A))$ holds in general. That is why, for $X, Y \in \{\text{DFA}, \text{MDFA}, \text{NFA}, \text{MNFA}, \lambda\text{-NFA}, \lambda\text{-MNFA}\}$, the family of

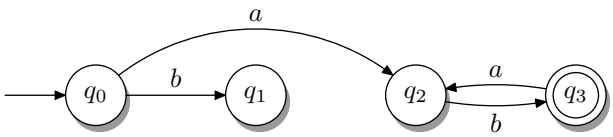

Fig. 1. The DFA A with $L(A) = (ab)^+$ interpreted as a jumping finite automaton accepts the language $L_J(A) = \{\, w \in \{a, b\}^+ \mid |w|_a = |w|_b \,\}$.

languages accepted by automata of type X interpreted as jumping finite automata equals the family of languages accepted by automata of type Y interpreted as jumping finite automata. This language family is referred to as **JFA**. Clearly, we have **JFA** = **pJFA** and it obeys a nice characterization.

Theorem 2. *A language L is in* **JFA** *if and only if L is permutation closed and semi-linear.* □

Moreover **pCF** ⊂ **JFA** ⊂ **pCS** and **JFA** strictly contains **pROWJ**; here **ROWJ** refers to the family of languages that are accepted by right one-way finite jumping automata, an automaton model introduced in the next subsection—in addition **JFA** is incomparable to the families **REG**, **DCF**, **CF**, **ROWJ**; for these results see, e.g., [1, 4, 5, 9].

Let us take a look at the descriptional complexity of these devices. It is obvious that the standard automata construction from the theory of finite state devices also applies to our setting whenever the underlying machine is interpreted as a jumping finite automaton. Hence, the conversion of a type X automaton, for $X \in \{\text{MDFA}, \text{MNFA}, \lambda\text{-NFA}, \lambda\text{-MNFA}\}$, to an equivalent NFA device (both machines interpreted as jumping automata) results in state complexity increase of at most one state, which is used to simulate the union of languages for automata of types from $\{\text{MDFA}, \text{MNFA}, \lambda\text{-MNFA}\}$. Removing spontaneous transitions does not increase the number of states at all. The remaining conversion from an NFA to a DFA is at most 2^n, where n is the number of states of the NFA, due to the powerset construction that can be successfully applied in our setting, too. Since the equivalence needs to be w.r.t. the Parikh image of the languages one can prove a better conversion bound. Namely, in [8] it was shown that for each NFA with n states a Parikh equivalent DFA with $e^{O(\sqrt{n \log n})}$ states can be constructed and that this bound is asymptotically tight. Thus, in our setting this result reads as follows:

Theorem 3. *Let A be an n-state NFA. Then $e^{O(\sqrt{n \log n})}$ states are sufficient and necessary in the worst for a DFA B to accept the language $L_J(A)$ if B is interpreted as a jumping finite automaton, that is, $L_J(B) = L_J(A)$.* □

3.1 Right One-Way Jumping Finite Automata

We deal with a variant of jumping finite automata, namely so called right one-way jumping finite automata, which were introduced in [4] in the deterministic

case. We now generalize the definition of these devices to nondeterministic machines without λ-transitions. Their behaviour can be described as follows. The read head starts the computation at the leftmost symbol of the input and moves to the right. If $R(p,a) \neq \emptyset$ for the current state p and the next input symbol a, one of the possible transitions is executed, the symbol a is consumed, and the read head moves on to the next symbol. If, however, $R(p,a) = \emptyset$, the read head jumps over a, does not consume it, and moves on to the next symbol. When the head reaches the end of the input, it jumps back to the beginning and continues the computation, which goes on until no input symbol is left.

A formal definition is given in the following: let $A = (Q, \Sigma, R, S, F)$ be an MNFA. The *right one-way jumping relation*, symbolically denoted by $\circlearrowright_A$, over $Q\Sigma^*$ is defined as follows. Let $p, q \in Q$, $a \in \Sigma$, $w \in \Sigma^*$, and $q \in R(p,a)$. Then, we have $paw \circlearrowright_A qw$. Now, let $p \in Q$, $a \in \Sigma$, and $w \in \Sigma^*$ with $R(p,a) = \emptyset$. In this case we get $paw \circlearrowright_A pwa$. We also write $\circlearrowright$ instead of $\circlearrowright_A$ if it is clear which automaton we are referring to. The *language accepted by A interpreted as a right one-way jumping finite automaton* is

$$L_R(A) = \{\, w \in \Sigma^* \mid \exists s \in S, f \in F : sw \circlearrowright^* f \,\}.$$

In order to keep the presentation simple, we speak of a ROWJFA, MROWJFA, NROWJFA, and MNROWJFA, respectively, if we interpret a DFA, MDFA, NFA, and a MNFA as a right one-way device, respectively. The corresponding language families are referred to as **ROWJ**, **MROWJ**, **NROWJ**, and **MNROWJ**, resp. Clearly, for unary alphabets all these families correspond with **REG**.

Example 4. We continue our previous example. To show how ROWJFAs work, we give an example computation of A, interpreted as an ROWJFA, on the input word $aabbba$:

$$q_0aabbba \circlearrowright q_2abbba \circlearrowright^2 q_3bbaa \circlearrowright^3 q_2abb \circlearrowright^2 q_3ba \circlearrowright^2 q_2b \circlearrowright q_3$$

That shows $aabbba \in L_R(A)$. Analogously, one can see that every word that contains the same number of a's and b's and that begins with an a is in $L_R(A)$. On the other hand, no other word can be accepted by A, interpreted as an ROWJFA. So, we get $L_R(A) = \{\, w \in a\{a,b\}^* \mid |w|_a = |w|_b \,\}$. Notice that this language is non-regular and not closed under permutation. □

Now the question arises how the right one-way jumping relation can be generalized to work on automata with λ-transitions? Sometimes we call these transitions also *spontaneous* transitions. To cope with spontaneous transitions we have three different interpretation possibilities for the right one-way jumping relation—as above we speak of λ-NROWJFAs (λ-MNROWJFAs, respectively), if λ-NFAs (λ-MNFAs, respectively) are interpreted as right one-way jumping devices:

Type 1. The *first type of right one-way jumping finite automata* is only allowed to jump over an input symbol, if A cannot perform a λ-transition in the current state. So, the *right one-way jumping relation of type 1*, symbolically

denoted by $\circlearrowright_{A,1}$, over $Q\Sigma^*$ is defined as follows. Let $p, q \in Q$, $a \in \Sigma \cup \{\lambda\}$, $w \in \Sigma^*$, and $q \in R(p, a)$. Then, we have $paw \circlearrowright_{A,1} qw$. Now, let $p \in Q$, $a \in \Sigma$, and $w \in \Sigma^*$ with $R(p, a) = \emptyset = R(p, \lambda)$. In this case it holds $paw \circlearrowright_{A,1} pwa$.

Type 2. If the *second type of right one-way jumping finite automata* has no transition for the current state and the next input symbol, but can perform a λ-transition in the current state, the automaton is allowed to choose if it performs a jump or if it uses a λ-transition. However, if the automaton decides to jump, it has to jump over all the following input symbols that cannot be read in the current state and must read the next input symbol that can be read in the current state. Not until now, the automaton is allowed to perform a λ-transition again. So, we set

$$\Sigma_{R,p} = \{\, a \in \Sigma \mid R(p, a) \neq \emptyset \,\}$$

for $p \in Q$. Therefore the *right one-way jumping relation of type 2*, symbolically denoted by $\circlearrowright_{A,2}$, over $Q\Sigma^*$ is defined as follows. For $p, q \in Q$, $w \in \Sigma^*$, and $q \in R(p, \lambda)$ it holds $pw \circlearrowright_{A,2} qw$. Now, let $p, q \in Q$, $a \in \Sigma$, $v \in (\Sigma \setminus \Sigma_{R,p})^*$, and $w \in \Sigma^*$ with $q \in R(p, a)$. Then, we get $pvaw \circlearrowright_{A,2} qwv$.

Type 3. If the *third type of right one-way jumping finite automata* has no transition for the current state and the next input symbol, but can perform a λ-transition in the current state, the automaton is allowed to choose if it jumps over the next input symbol or if it uses a λ-transition. Thus, the *right one-way jumping relation of type 3*, symbolically denoted by $\circlearrowright_{A,3}$, over $Q\Sigma^*$ is defined as follows. Let $p, q \in Q$, $a \in \Sigma \cup \{\lambda\}$, $w \in \Sigma^*$, and $q \in R(p, a)$. In this case we have $paw \circlearrowright_{A,3} qw$. For $p \in Q$, $a \in \Sigma$, and $w \in \Sigma^*$ with $R(p, a) = \emptyset$ it holds $paw \circlearrowright_{A,3} pwa$.

Let $i \in \{1, 2, 3\}$. We also write $\circlearrowright_i$ instead of $\circlearrowright_{A,i}$ if it is clear which automaton we are referring to. The *language accepted by A interpreted as a right one-way jumping finite automaton of type i* is defined to be

$$L_{Ri}(A) = \{\, w \in \Sigma^* \mid \exists s \in S, f \in F : sw \circlearrowright_i^* f \,\}.$$

This give rise to the language families $\lambda_{\mathbf{i}}\mathbf{NROWJ}$ ($\lambda_{\mathbf{i}}\mathbf{MNROWJ}$, respectively) of all languages accepted by λ-NROWJFAs (λ-MNROWJFAs, respectively) of type i, for $1 \leq i \leq 3$. Again, all these families correspond with **REG** for unary alphabets.

Example 5. Let $A = (\{q_0, q_1, \ldots, q_5\}, \{a, b, c\}, R, \{q_0\}, \{q_5\})$ be the λ-NROWJFA with

$$R = \{q_0\lambda \to q_1, q_0a \to q_4, q_1a \to q_2, q_1b \to q_2, \\ q_1c \to q_3, q_3a \to q_4, q_4b \to q_5, q_5a \to q_4\}.$$

The automaton A is depicted in Figure 2.

It is easy to see that that $L(A) = \{\lambda, c\}(ab)^+$. Let L refer to the language $\{\, w \in \{a, b\}^+ \mid |w|_a = |w|_b \,\}$. For the three interpretations on how to cope with spontaneous transitions we get the following languages—let letter $x \in \{a, b\}$, words $u, v \in \{b, c\}^*$, and $w \in \{a, b, c\}^*$:

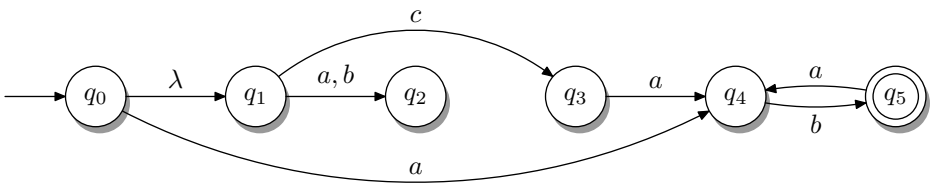

Fig. 2. The λ-NROWJFA A with $L(A) \subset L_{R1}(A) \subset L_{R2}(A) \subset L_{R3}(A) \subset L_J(A)$.

– In type 1 we have the computations $q_0aw \circlearrowleft_1 q_4w$, $q_0xw \circlearrowleft_1^2 q_2w$, and $q_0cvaw \circlearrowleft_1^2 q_3vaw \circlearrowleft_1^{|v|+1} q_4wv$. This implies

$$L_{R1}(A) = \{\, w \in a\{a,b\}^* \mid |w|_a = |w|_b \,\} \cup cL.$$

– For type 2 we get the behaviour $q_0vaw \circlearrowleft_2 q_4wv$, $q_0xw \circlearrowleft_2^2 q_2w$, and finally $q_0cvaw \circlearrowleft_2^2 q_3vaw \circlearrowleft_2 q_4wv$. That gives us

$$L_{R2}(A) = \{\lambda, c\}L.$$

– In type 3 it holds $q_0vaw \circlearrowleft_3^{|v|+1} q_4wv$, $q_0vxw \circlearrowleft_3^{|v|+2} q_2wv$, and moreover $q_0vcuaw \circlearrowleft_3^{|v|+2} q_3uawv \circlearrowleft_3^{|u|+1} q_4wvu$. Hence, we have

$$L_{R3}(A) = L \cup \left\{\, b^ncw \;\middle|\; n \geq 0, w \in \{a,b\}^+, |b^ncw|_a = |b^ncw|_b \,\right\}.$$

Furthermore, we get

$$L_J(A) = \mathsf{perm}(L(A)) = \{\, w \in \{a,b,c\}^* \mid |w|_a = |w|_b > 0, |w|_c \in \{0,1\} \,\}.$$

It follows

$$L(A) \subseteq L_{R1}(A) \subseteq L_{R2}(A) \subseteq L_{R3}(A) \subseteq L_J(A)$$

where the inclusions are strict. This inclusion chain is not a coincidence, as we will see next. □

From our definitions, we can deduce:

Lemma 6. *Let $A = (Q, \Sigma, R, S, F)$ be a λ-MNROWJFA. Then, the following holds:*

1. *We have $L(A) \subseteq L_{R1}(A) \subseteq L_{R2}(A) \subseteq L_{R3}(A) \subseteq L_J(A)$ and these languages are semi-linear.*
2. *If A is an MNROWJFA, then $L_R(A) = L_{Ri}(A)$, for $1 \leq i \leq 3$. Hence the inclusion chain $L(A) \subseteq L_R(A) \subseteq L_J(A)$ applies.*
3. *If $R(p,a) \neq \emptyset$, for every state $p \in Q$ and letter $a \in \Sigma$, then $L(A) = L_{Ri}(A)$, for $1 \leq i \leq 3$.* □

Lemma 6 implies that every regular language is accepted by a ROWJFA, because every regular language is accepted by a DFA with total transition function. On the other hand, we have seen in our example that ROWJFAs can accept non-regular languages. Indeed, it was already shown in [4] that **REG** $\subset$ **ROWJ**.

3.2 Variants of Right One-Way Jumping Finite Automata Without Spontaneous Transitions

From the language families **ROWJ**, **MROWJ**, **NROWJ**, and **MNROWJ** and their permutation closed variants the former two were already investigated in the literature. For inclusion relations and incomparability results see, e.g., [1, 2]. Clearly, **MROWJ** consists exactly of the finite unions of languages from **ROWJ**. However, it is not clear that every language from **pMROWJ** is a finite union of languages from **pROWJ**. Nice characterizations for both language families were obtained. More precisely, a permutation closed language L is in **ROWJ** if and only if the Myhill-Nerode relation $\sim_L$ has only a finite number of positive equivalence classes [1]. In [2], concerning the language family **MROWJ** it was shown that for a permutation closed language L the following statements are equivalent: (1) For all $w \in \Sigma^*$, the language $L/^d w$ is in **pMROWJ**, where

$$L/^d w = \{\, v \in \Sigma^* \mid vw \in L \text{ and } (|v|_a = 0 \vee |w|_a = 0), \text{ for every } a \in \Sigma \,\}$$

is the *disjoint quotient* of a language $L \subseteq \Sigma^*$ by a word $w \in \Sigma^*$, and (2) there is an $n \geq 0$ and permutation closed languages $L_1, L_2, \ldots, L_n \subseteq \Sigma^*$ such that $L = \bigcup_{i=1}^{n} L_i$ and for all $i \in \{1, 2, \ldots, n\}$ the language L_i has only a finite number of positive Myhill-Nerode equivalence classes. Both characterization results are based on a relation between languages accepted by ROWJFAs and MROWJFAs and semi-linear sets. For our language families we find the following situation.

Theorem 7. *The following strict inclusions of language families hold:*

1. **ROWJ** $\subset$ **MROWJ** *and* **ROWJ** $\subset$ **NROWJ**.
2. **MROWJ** $\subset$ **MNROWJ** *and* **NROWJ** $\subset$ **MNROWJ**. □

The languages used in the proof of the previous theorem show that the language families **MROWJ** and **NROWJ** are incomparable.

Corollary 8. *The language families* **MROWJ** *and* **NROWJ** *are incomparable.* □

Before we investigate the hierarchy of permutation closed language families we first state a result that shows that whenever the underlying device is nondeterministic and accepts a permutation closed language it does not matter whether the automaton has multiple initial states or not. Moreover, we show that the induced language family coincides with **JFA**, the family of languages accepted by jumping finite automata.

Theorem 9. *We have* **pNROWJ** = **pMNROWJ** = **JFA**. □

From [2], we know **pROWJ** $\subset$ **pMROWJ**. Furthermore, the permutation closed language $\{\, w \in \{a, b\}^* \mid |w|_a \neq |w|_b \,\}$ is clearly in **JFA**, but was shown to be not in **pMROWJ** in [2]. Hence, together with the previous theorem we get:

Theorem 10. *It holds* **pROWJ** $\subset$ **pMROWJ** $\subset$ **pNROWJ** = **pMNROWJ**. □

3.3 Variants of Right One-Way Jumping Finite Automata With Spontaneous Transitions

In this subsection we study λ-NROWJFAs and λ-MNROWJFAs under the three semantics introduced earlier. Observe that by definition we have the inclusions $\mathbf{MNROWJ} \subseteq \lambda_\mathbf{i}\mathbf{MNROWJ}$ and $\mathbf{NROWJ} \subseteq \lambda_\mathbf{i}\mathbf{NROWJ} \subseteq \lambda_\mathbf{i}\mathbf{MNROWJ}$, for every $i \in \{1, 2, 3\}$. First we show that λ-NROWJFAs and λ-MNROWJFAs have the same computational power, if we fix the semantics.

Lemma 11. *For $i \in \{1, 2, 3\}$, we have $\lambda_\mathbf{i}\mathbf{NROWJ} = \lambda_\mathbf{i}\mathbf{MNROWJ}$.* □

Next we consider the case when the semantics on λ-MNROWJFAs varies. We show that λ-MNROWJFAs of type 1 are at most as powerful as those of type 2 and those of type 3:

Lemma 12. *We have $\lambda_\mathbf{1}\mathbf{MNROWJ} \subseteq \lambda_\mathbf{2}\mathbf{MNROWJ}$ and $\lambda_\mathbf{1}\mathbf{MNROWJ} \subseteq \lambda_\mathbf{3}\mathbf{MNROWJ}$.* □

Our next result shows that MNROWJFAs have indeed the same power as λ-MNROWJFAs of type 2.

Lemma 13. *We have $\mathbf{MNROWJ} = \lambda_\mathbf{2}\mathbf{MNROWJ}$.* □

By definition, it holds $\mathbf{MNROWJ} \subseteq \lambda_\mathbf{1}\mathbf{MNROWJ}$. Thus, the previous three Lemmas 11, 12, and 13 give us the following equalities of language families.

Corollary 14. *We have $\mathbf{MNROWJ} = \lambda_\mathbf{i}\mathbf{NROWJ} = \lambda_\mathbf{i}\mathbf{MNROWJ}$, for i with $i \in \{1, 2\}$.* □

In contrast to the last result, λ-NROWJFAs of type 3 are more powerful than MNROWJFAs.

Theorem 15. *It holds $\mathbf{MNROWJ} \subset \lambda_\mathbf{3}\mathbf{NROWJ}$.* □

Our language families are all included in the complexity classes $\mathbf{NTIME}(n^2)$ and in $\mathbf{NSPACE}(n)$:

Theorem 16. *For $i \in \{1, 2, 3\}$, the family $\lambda_\mathbf{i}\mathbf{MNROWJ}$ is properly included in $\mathbf{NTIME}(n^2)$ and in $\mathbf{NSPACE}(n)$.* □

For a λ-MNROWJFA A and $i \in \{1, 2, 3\}$ the language $L_{Ri}(A)$ contains a Parikh-equivalent regular sublanguage, because of Lemma 6. The only Parikh-equivalent sublanguage of a letter-bounded language is the language itself, so we get the following.

Corollary 17. *The letter-bounded languages contained in $\lambda_\mathbf{3}\mathbf{MNROWJ}$ are exactly the regular letter-bounded languages.* □

As a consequence, the language $\{ a^n b^n \mid n \geq 0 \}$ is not in $\lambda_\mathbf{3}\mathbf{MNROWJ}$. On the other hand, it was shown in [4] that $\mathbf{ROWJ}$ contains non-context-free languages. It follows:

Corollary 18. *Let $F \in \{\mathbf{ROWJ}, \lambda_\mathbf{3}\mathbf{MNROWJ}\}$ and $G \in \{\mathbf{DCF}, \mathbf{CF}\}$. Then, the language families F and G are incomparable.* □

4 Relations to Finite-State Acceptors With Translucent Letters

Finite-state acceptors with translucent letters were defined in [10]. For these devices, depending on the current internal state, some letters of the input alphabet are translucent, which means that the automaton does not see them and reads the first letter which is not translucent. So, their behaviour is similar to ROWJFAs, which jump over some letters of the input alphabet, also depending on the current state of the automaton. We will investigate the inclusion relations between the language families induced by finite-state acceptors with translucent letters and those families which come from ROWJFAs and their nondeterministic variants.

A *finite-state acceptor with translucent letters*, a NFAwtl for short, is a tuple $A = (Q, \Sigma, \tau, S, F, \delta)$, where Q is the finite *set of states*, Σ is the finite *input alphabet*, $\tau : Q \to 2^{\Sigma}$ is the *translucency mapping*, $S \subseteq Q$ is the set of *start states*, $F \subseteq Q$ is the set of *final states*, and $\delta : Q \times \Sigma \to 2^{Q}$ is the *transition function*. For each state $q \in Q$ the letters from $\tau(q)$ are *translucent in state* q. The automaton A is called *deterministic*, a DFAwtl for short, if $|S| = 1$ and $|\delta(q, a)| \leq 1$, for all $(q, a) \in Q \times \Sigma$.

A *configuration* of A is a string in $Q\Sigma^*$. The automaton A makes a *transition* from configuration $pvaw$ to configuration qvw if $q \in \delta(p, a)$, where $p, q \in Q$, $v \in (\tau(p))^*$, $a \in \Sigma \setminus \tau(p)$, and $w \in \Sigma^*$. We denote this by $pvaw \vdash_A qvw$ or just $pvaw \vdash qvw$ if it is clear which automaton we are referring to. Then, the *language accepted by* A is

$$L(A) = \{\, w \in \Sigma^* \mid \exists s \in S, f \in F, v \in (\tau(f))^* : sw \vdash^* fv \,\}.$$

We say that A *accepts* $w \in \Sigma^*$ if $w \in L(A)$ and that A *rejects* w otherwise. The family of all languages accepted by a NFAwtl (DFAwtl, respectively) will be denoted by **NTRANS** (**DTRANS**, respectively). These families are incomparable to the language families induced by ROWJFAs and their nondeterministic variants:

Theorem 19. *Consider language families* $F \in \{\mathbf{DTRANS}, \mathbf{NTRANS}\}$ *and* $G \in \{\mathbf{ROWJ}, \lambda_{\mathbf{3}}\mathbf{NROWJ}\}$. *Then, the families* F *and* G *are incomparable.* □

From [10] it is known that **pNTRANS** = **JFA**. The family **pDTRANS** has the following inclusion relations to **pROWJ** and **pMROWJ**.

Theorem 20. *We have* **pROWJ** $\subset$ **pDTRANS**. *The families* **pDTRANS** *and* **pMROWJ** *are incomparable.* □

In [10] it was shown that for each NFAwtl A there is an NFAwtl $B = (Q, \Sigma, \tau, S, F, \delta)$ with $L(A) = L(B)$ and for all $w \in \Sigma^*$, $s \in S$, $f \in F$, and $v \in (\tau(f))^*$ with $sw \vdash_B^* fv$ it holds $v = \lambda$. It was stated as an open problem if this property also holds for DFAswtl. We can solve this open problem:

Corollary 21. *There is a DFAwtl* A *such that there does* not *exist a DFAwtl* $B = (Q, \Sigma, \tau, \{s\}, F, \delta)$ *with* $L(A) = L(B)$ *and for all* $w \in \Sigma^*$, $f \in F$, *and* $v \in (\tau(f))^*$ *with* $sw \vdash_B^* fv$ *it holds* $v = \lambda$. □

5 Conclusions

We have investigated nondeterministic variants of ROWJFAs and showed inclusion and incomparability results of the induced language families. In order to complete the picture of these new language families, it remains to study closure properties and decision problems for these devices. Since languages accepted by variants of jumping automata are all semi-linear, it is worth to consider their relation to Petri Nets, since a wide variety of them enjoy semi-linear reachability sets—see, e.g., [12].

References

1. Beier, S., Holzer, M.: Properties of right one-way jumping finite automata. In: Konstantinidis, S., Pighizzini, G. (eds.) Proceedings of the 20th International Workshop on Descriptional Complexity of Formal Systems. pp. 11–23. No. 10952 in LNCS, Springer, Halifax, Nova Scotia, Canada (2018)
2. Beier, S., Holzer, M.: Semi-linear lattices and right one-way jumping finite automata (extended abstract). In: Hospodár, M., Jirásková, G. (eds.) Proceedings of the 24th International Conference on Implementation and Application of Automata. LNCS, Springer, Košice, Slovakia (2019), to appear
3. Beier, S., Holzer, M., Kutrib, M.: Operational state complexity and decidability of jumping finite automata. In: Charlier, É., Leroy, J., Rigo, M. (eds.) Proceedings of the 21st International Conference on Developments in Language Theory. pp. 96–108. No. 10396 in LNCS, Springer, Liège, Belgium (2017)
4. Chigahara, H., Fazekas, S., Yamamura, A.: One-way jumping finite automata. Internat. J. Found. Comput. Sci. **27**(3), 391–405 (2016)
5. Fernau, H., Paramasivan, M., Schmid, M.L.: Jumping finite automata: Characterizations and complexity. In: Drewes, F. (ed.) Proceedings of the 20th Conference on Implementation and Application of Automata. pp. 89–101. No. 9223 in LNCS, Springer, Umeå, Sweden (2015)
6. Fernau, H., Paramasivan, M., Schmid, M.L., Vorel, V.: Characterization and complexity results on jumping finite automata. Theoret. Comput. Sci. **679**, 31–52 (2017)
7. Harrison, M.A.: Introduction to Formal Language Theory. Addison-Wesley (1978)
8. Lavado, G.J., Pighizzini, G., Seki, S.: Operational state complexity under Parikh equivalence. In: Jürgensen, H., Karhumäki, J., Okhotin, A. (eds.) Proceedings of the 16th International Workshop on Descriptional Complexity of Formal Systems. pp. 294–305. No. 8614 in LNCS, Springer, Turku, Finland (2014)
9. Meduna, A., Zemek, P.: Jumping finite automata. Internat. J. Found. Comput. Sci. **23**(7), 1555–1578 (2012)
10. Nagy, B., Otto, F.: Finite-state acceptors with translucent letters. In: Bel-Enguix, G., Dahl, V., De La Puente, A.O. (eds.) 1st Workshop on AI Methods for Interdisciplinary Research in Language and Biology—BILC. pp. 3–13. SciTePress, Rome, Italy (2011)
11. Vorel, V.: On basic properties of jumping finite automata. Internat. J. Found. Comput. Sci. **29**(1), 1–16 (2018)
12. Yen, H.C.: Petri nets and semilinear sets (extended abstract). In: Sampaio, A., Wang, F. (eds.) Proceedings of the 13th International Colloquium on Theoretical Aspects of Computing. pp. 25–29. No. 8587 in LNCS, Springer, Taipei, Taiwan (2016)