RAIRO-Theor. Inf. Appl. 57 (2023) 8
https://doi.org/10.1051/ita/2023009

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

NON-RETURNING DETERMINISTIC AND NONDETERMINISTIC FINITE AUTOMATA WITH TRANSLUCENT LETTERS

František Mráz[1] and Friedrich Otto[2,*]

Abstract. Here, we propose a variant of the nondeterministic finite automaton with translucent letters (NFAwtl), which, after reading and deleting a letter, does not return to the left end of its tape, but instead continues from the position of the letter just deleted. When the end-of-tape marker is reached, our automaton can decide whether to accept, reject, or continue, which means that it reads the remaining tape contents again from the beginning. This type of automaton, called a *non-returning finite automaton with translucent letters* or an *nrNFAwtl*, is strictly more expressive than the NFAwtl. We study the expressive capacity of this type of automaton and that of its deterministic variant, comparing them to various other types of finite automata that do not simply read their input from left to right. Also, we are interested in the closure properties of the resulting classes of languages and in the complexity of the fixed membership problem.

Mathematics Subject Classification. 68Q45.

Received January 28, 2023. Accepted September 19, 2023.

1. Introduction

While a (deterministic or nondeterministic) finite automaton reads its input strictly from left to right, letter by letter, by now, many types of automata have been considered in the literature that process their inputs in a different, more involved way. In this aspect, the most extreme is the *jumping finite automaton* of Meduna and Zemek [11] (see also [8]), which, after reading a letter, jumps to an arbitrary position of the remaining input. It is known that the jumping finite automaton accepts languages that are not even context-free, like the language $\{ w \in \{a,b,c\}^* \mid |w|_a = |w|_b = |w|_c \}$, but at the same time, it does not even accept the finite language $\{ab\}$.

Another example is the *nondeterministic linear automaton* (or NLA) studied by Loukanova in [10], which is a nondeterministic finite automaton with two heads, one reading the input from left to right, the other reading the input from right to left. This model can be simulated by a model with one head that reads the first and the last letter alternatingly. It is easily seen that this model characterizes the class LIN of linear context-free languages. Actually, the NLA corresponds to the $5' \to 3'$-*sensing Watson-Crick automaton* defined by Nagy in [15].

Moreover, there is the *restarting automaton* as introduced by Jančar, Mráz, Plátek, and Vogel in [9], which processes a given input in cycles, in each cycle scanning the remaining input from left to right until it deletes

Keywords and phrases: Finite automaton, translucent letter, language class.

[1] Charles University, Faculty of Mathematics and Physics, Malostranské nám. 25, 118 25 Prague 1, Czech Republic.
[2] Fachbereich Elektrotechnik/Informatik, Universität Kassel, 34109 Kassel, Germany.

* Corresponding author: f.otto@uni-kassel.de

one or more letters, returns its read/write window to the left end of the remaining input, and reenters its initial state. When using a window of size larger than one, these so-called R-automata accept a proper superclass of the regular languages that is incomparable to the context-free and the growing context-sensitive languages (see, *e.g.*, [22]), while with a window of size one, they accept exactly the regular languages [12].

Finally, there is the (deterministic and nondeterministic) finite automaton *with translucent letters* (or DFAwtl and NFAwtl) of Nagy and Otto [18], which is equivalent to a cooperating distributed system of stateless deterministic R-automata with windows of size one. For each state q of an NFAwtl, there is a set $\tau(q)$ of *translucent letters*, which is a subset of the input alphabet containing those letters that the automaton cannot see when it is in state q. Accordingly, in each step, the NFAwtl just reads (and deletes) the first letter from the left, which it can see, that is, the first letter that is not translucent in its current state. It has been proved that the NFAwtl accepts a class of semi-linear languages that properly contains all rational trace languages, while its deterministic variant, the DFAwtl, is properly less expressive. In fact, the DFAwtl just accepts a class of languages that is incomparable to the rational trace languages with respect to inclusion [17, 19–21]. In addition, while the obvious upper bound for the time complexity of the membership problem for a DFAwtl is $\mathrm{DTIME}(n^2)$, a better upper bound of $\mathrm{DTIME}(n \cdot \log n)$ is derived in [16].

Here, we propose a variant of the nondeterministic finite automaton with translucent letters, which, after reading and deleting a letter, does not return its head to the left end of its tape, but instead continues from the position of the letter just deleted. When the end-of-tape marker is reached, our automaton can decide whether to accept, reject, or continue, which means that it reads the remaining tape contents again from the beginning. We prove that this type of automaton, called a *non-returning finite automaton with translucent letters* or an *nrNFAwtl*, is strictly more expressive than the NFAwtl. However, as we shall see, its deterministic variant, the *nrDFAwtl*, which is more expressive than the DFAwtl, is still not powerful enough to accept all rational trace languages. In this paper, we concentrate on the problem of determining just how expressive these types of automata are and on the complexity of their fixed membership problem, but we are also interested in the closure and non-closure properties of the resulting classes of languages.

This paper is structured as follows. In Section 2, we present the formal definition of the non-returning finite automaton with translucent letters, explain its workings by a detailed example, and derive a kind of normalized form for this type of automaton. In the next section, we compare the classes of languages accepted by the nondeterministic and the deterministic non-returning finite automaton with translucent letters to the language classes accepted by the DFAwtl and the NFAwtl and to the rational trace languages, establishing some proper inclusion results and some incomparability results. Also, we present a few closure and non-closure properties for the classes of languages that are accepted by the nondeterministic and the deterministic non-returning finite automaton with translucent letters. Then, in Section 4, we compare the non-returning finite automaton with translucent letters to the jumping finite automaton, the right one-way jumping finite automaton of [2, 5], and the right-revolving finite automaton of [3], deriving the complete taxonomy of the resulting classes of languages. As it turns out, the non-returning DFAwtl can be seen as an extension of the right one-way jumping finite automaton that can detect the end of its input. Finally, in Section 5, we study the complexity of the fixed membership problem for the nrDFAwtl, showing that it is decidable in time $\mathrm{O}(n \cdot (\log n)^2)$ whether a word of length n is accepted by a given nrDFAwtl. In the final section, we summarize our results and state a number of open problems for future work.

2. Definitions

Throughout the paper, λ denotes the empty word, and the concatenation of two words u and v over an alphabet Σ is denoted by $u \cdot v$ or simply by uv.

The nondeterministic finite automaton with translucent letters was introduced in [18]. It is defined as follows.

Definition 2.1. A *finite automaton with translucent letters*, an *NFAwtl* for short, is defined as a 7-tuple $A = (Q, \Sigma, \triangleleft, \tau, I, F, \delta)$, where Q is a finite set of internal states, Σ is a finite alphabet of input letters, $\triangleleft \notin \Sigma$ is a special symbol that is used as an *end-of-tape marker*, $\tau : Q \to \mathcal{P}(\Sigma)$ is a *translucency mapping*, $I \subseteq Q$ is

a set of initial states, $F \subseteq Q$ is a set of final states, and $\delta : Q \times \Sigma \to \mathcal{P}(Q)$ is a *transition relation*. Here, it is required that, for each state $q \in Q$ and each letter $a \in \Sigma$, if $a \in \tau(q)$, then $\delta(q, a) = \emptyset$. For each state $q \in Q$, the letters from the set $\tau(q)$ are *translucent* for q, that is, in state q, the automaton A does not see these letters.

An NFAwtl $A = (Q, \Sigma, \lhd, \tau, I, F, \delta)$ works as follows. For an input word $w \in \Sigma^*$, it starts in a nondeterministically chosen initial state $q_0 \in I$ with the word $w \cdot \lhd$ on its tape. Assume that $w = a_1 a_2 \cdots a_n$ for some $n \geq 1$ and $a_1, a_2, \ldots, a_n \in \Sigma$, and assume that A is in state $q \in Q$. Then A looks for the first occurrence from the left of a letter that is not translucent for state q, that is, if $w = uav$ such that $u \in (\tau(q))^*$, $v \in \Sigma^*$, and $a \in (\Sigma \setminus \tau(q))$, then A nondeterministically chooses a state $q_1 \in \delta(q, a)$, erases the letter a from the tape, thus producing the tape contents $uv \cdot \lhd$, its internal state is set to q_1, the head returns to the first letter on the tape, and the computation continues. In case $\delta(q, a) = \emptyset$, A halts without accepting. Finally, if $w \in (\tau(q))^*$, then A sees the end-of-tape marker $\lhd$ and the computation halts. In this case, A accepts if q is a final state; otherwise, it does not accept. Thus, A executes the following computation relation on its set $Q \cdot \Sigma^* \cdot \{\lhd\} \cup \{\mathsf{Accept}, \mathsf{Reject}\}$ of configurations, where u, v, w are words over Σ and a is a letter from Σ:

$$qw \cdot \lhd \vdash_A \begin{cases} q'uv \cdot \lhd, & \text{if } w = uav,\ u \in (\tau(q))^*,\ a \notin \tau(q),\ \text{and } q' \in \delta(q, a), \\ \mathsf{Reject}, & \text{if } w = uav,\ u \in (\tau(q))^*,\ a \notin \tau(q),\ \text{and } \delta(q, a) = \emptyset, \\ \mathsf{Accept}, & \text{if } w \in (\tau(q))^* \text{ and } q \in F, \\ \mathsf{Reject}, & \text{if } w \in (\tau(q))^* \text{ and } q \notin F. \end{cases}$$

Observe that this definition also applies to configurations of the form $q \cdot \lhd$, that is, $q \cdot \lambda \cdot \lhd \vdash_A \mathsf{Accept}$ holds if and only if q is a final state. A word $w \in \Sigma^*$ is *accepted by* A if there exists an initial state $q_0 \in I$ such that $q_0 w \cdot \lhd \vdash_A^* \mathsf{Accept}$, where $\vdash_A^*$ denotes the reflexive and transitive closure of the single-step computation relation $\vdash_A$. Now $L(A) = \{\, w \in \Sigma^* \mid w \text{ is accepted by } A \,\}$ is the *language accepted by* A and $\mathcal{L}(\mathsf{NFAwtl})$ denotes the class of all languages that are accepted by NFAwtls.

Definition 2.2. An NFAwtl $A = (Q, \Sigma, \lhd, \tau, I, F, \delta)$ is a *deterministic finite automaton with translucent letters*, abbreviated as *DFAwtl*, if $|I| = 1$ and if $|\delta(q, a)| \leq 1$ for all $q \in Q$ and all $a \in \Sigma$. Then $\mathcal{L}(\mathsf{DFAwtl})$ denotes the class of all languages that are accepted by DFAwtls.

To emphasize the distinction between deterministic and nondeterministic finite automata with translucent letters, if an NFAwtl $A = (Q, \Sigma, \lhd, \tau, I, F, \delta)$ with $I = \{q_0\}$ is a DFAwtl, then we denote it simply as $A = (Q, \Sigma, \lhd, \tau, q_0, F, \delta)$. In addition, for any states $q, q' \in Q$ and any letter $a \in \Sigma$, we write $\delta(q, a) = q'$ instead of $\delta(q, a) = \{q'\}$ and we say that $\delta(q, a)$ is undefined if $\delta(q, a) = \emptyset$. For future reference, we present an example of a DFAwtl.

Example 2.3. Let $A_c = (Q, \Sigma, \lhd, \tau, q_0, F, \delta)$ be the DFAwtl that is given through $Q = \{q_0, q_a, q'_a, q_b, q'_b\}$, $\Sigma = \{a, b, a', b'\}$, $F = \{q_0\}$, and the functions τ and δ that are defined as follows:

$$\begin{array}{llll} \tau(q_0) = \emptyset, & \tau(q_a) = \{a', b'\}, & \tau(q'_a) = \{a, b\}, & \\ & \tau(q_b) = \{a', b'\}, & \tau(q'_b) = \{a, b\}, & \\ \delta(q_0, a) = q'_a, & \delta(q_0, b) = q'_b, & \delta(q_0, a') = q_a, & \delta(q_0, b') = q_b, \\ \delta(q_a, a) = q_0, & \delta(q_b, b) = q_0, & \delta(q'_a, a') = q_0, & \delta(q'_b, b') = q_0, \end{array}$$

and δ is undefined for all other pairs from $Q \times \Sigma$. For the word $abba'b'ab'a'$, the DFAwtl A_c executes the following accepting computation:

$$\begin{array}{ccccccc} q_0 abba'b'ab'a' \cdot \lhd & \vdash_{A_c} & q'_a bba'b'ab'a' \cdot \lhd & \vdash_{A_c} & q_0 bbb'ab'a' \cdot \lhd & \vdash_{A_c} & q'_b bb'ab'a' \cdot \lhd \\ \vdash_{A_c} & q_0 bab'a' \cdot \lhd & \vdash_{A_c} & q'_b ab'a' \cdot \lhd & \vdash_{A_c} & q_0 aa' \cdot \lhd & \\ \vdash_{A_c} & q'_a a' \cdot \lhd & \vdash_{A_c} & q_0 \cdot \lhd & \vdash_{A_c} & \mathsf{Accept}. & \end{array}$$

In fact, if φ denotes the morphism that is defined through $\varphi(a) = a'$ and $\varphi(b) = b'$, then it is easily checked that $L(A_c) = \bigcup_{w \in \{a,b\}^*} \mathrm{sh}(w, \varphi(w))$, where sh denotes the shuffle operation defined for any pair of words u, v over an alphabet Σ as

$$\mathrm{sh}(u,v) = \{\, u_1 v_1 u_2 v_2 \cdots u_k v_k \mid k \geq 1, u_1, u_2, \ldots, u_k, v_1, v_2, \ldots, v_k \in \Sigma^* : u = u_1 u_2 \cdots u_k, v = v_1 v_2 \cdots v_k \}.$$

The above language $L(A_c)$ is not context-free. In fact, it is not even a growing context-sensitive language. Let $\pi : \{a, b, a', b'\}^* \to \{a, b\}^*$ be the morphism that is defined through $a \mapsto a$, $b \mapsto b$, $a' \mapsto a$, and $b' \mapsto b$. Then

$$\pi(L(A_c) \cap (\{a,b\}^* \cdot \{a',b'\}^*)) = \pi(\{\, w\varphi(w) \mid w \in \{a,b\}^* \,\}) = \{\, ww \mid w \in \{a,b\}^* \,\},$$

which is the copy language on $\{a, b\}^*$ that is not growing context-sensitive [4]. As the class GCSL of growing context-sensitive languages is closed under the operations of intersection with regular sets and non-erasing morphisms, this implies that the language $L(A_c)$ is not growing context-sensitive, either. Thus, Example 2.3 shows that DFAwtls already accept some quite complicated languages in comparison to the Chomsky hierarchy.

Recall that a language $L \subseteq \Sigma^*$ is a *rational trace language* if there exist a dependency relation D on Σ and a regular language $R \subseteq \Sigma^*$ such that L coincides with the closure of R under the partial commutativity relation $\equiv_D$ induced by D (see, *e.g.*, [7]). Here, a dependency relation is a binary relation on Σ that is reflexive and symmetric, and the partial commutativity relation $\equiv_D$ is the congruence on Σ^* that is induced by the pairs $\{\, (uabv, ubav) \mid a, b \in \Sigma, (a, b) \notin D, \text{ and } u, v \in \Sigma^* \,\}$.

Now let

$$D = \{(a,a), (a,b), (b,a), (b,b), (a',a'), (a',b'), (b',a'), (b',b')\}$$

be a dependency relation on $\Sigma = \{a, a', b, b'\}$. It is easily seen that the language $L(A_c)$ is the rational trace language that is obtained from the regular language $\{aa', bb'\}^*$ through the above dependency relation D [17, 20]. Hence, we obtain the following non-inclusion result.

Theorem 2.4. *The class $\mathcal{LRAT}$ of rational trace languages is not contained in the language class GCSL.*

The NFAwtl is equivalent to cooperating distributed systems of a very restricted type of deterministic restarting automata of window size one [18]. From this equivalence, it follows that the NFAwtls accept a class of semi-linear languages that properly contains the class of all rational trace languages. At the same time, DFAwtls are less expressive, as they just accept a class of languages that is incomparable to the rational trace languages with respect to inclusion [17, 19–21]. Furthermore, while the obvious upper bound for the time complexity of the fixed membership problem for a DFAwtl is $\mathrm{DTIME}(n^2)$, a better upper bound of $\mathrm{DTIME}(n \cdot \log n)$ is derived in [16], and a corresponding result holds for the nondeterministic case.

As defined above, an NFAwtl performs each step of its computation starting from the first letter on its tape: it looks for the first letter that is not translucent in its current state, deletes it and returns to the first letter. Here, we propose a variant of this type of automaton that does not return to the first letter, but continues from the position of the deleted letter, returning to the first letter only after the tape content has been read completely. Next, we present the formal definition of this type of automaton, which is called the *non-returning finite automaton with translucent letters* or *nrNFAwtl* for short.

Definition 2.5. An *nrNFAwtl* is defined by a 6-tuple $A = (Q, \Sigma, \lhd, \tau, I, \delta)$, where Q is a finite set of internal states, Σ is a finite alphabet of input letters, $\lhd \notin \Sigma$ is a special symbol that is used as an *end-of-tape marker*, $\tau : Q \to \mathcal{P}(\Sigma)$ is a *translucency mapping*, $I \subseteq Q$ is a set of initial states, and

$$\delta : Q \times (\Sigma \cup \{\lhd\}) \to (\mathcal{P}(Q) \cup \{\mathsf{Accept}\})$$

is a *transition relation*. Here it is required that, for each state $q \in Q$ and each letter $a \in \Sigma$, $\delta(q,a) \subseteq Q$, and if $a \in \tau(q)$, then $\delta(q,a) = \emptyset$. For each state $q \in Q$, the letters from the set $\tau(q)$ are *translucent* for q, that is, in state q, the automaton A does not see these letters.

An nrNFAwtl $A = (Q, \Sigma, \lhd, \tau, I, \delta)$ is a *non-returning deterministic finite automaton with translucent letters*, abbreviated as *nrDFAwtl*, if $|I| = 1$ and if $|\delta(q,a)| \leq 1$ for all $q \in Q$ and all $a \in \Sigma \cup \{\lhd\}$.

Similarly as for a DFAwtl, if $A = (Q, \Sigma, \lhd, \tau, I, \delta)$ is an nrNFAwtl that is deterministic, then we write $A = (Q, \Sigma, \lhd, \tau, q_0, \delta)$, where q_0 is the only initial state of A, and, for any states q, q' and any letter a, we write $\delta(q,a) = q'$ instead of $\delta(q,a) = \{q'\}$ and say that $\delta(q,a)$ is undefined if $\delta(q,a) = \emptyset$,

From the above definition, we see that $\delta(q, \lhd)$ is either a subset of Q or the operation Accept, that is, when seeing the end-of-tape marker $\lhd$ in state q, the nrNFAwtl A has either the option to change its state or to accept. The nrNFAwtl $A = (Q, \Sigma, \lhd, \tau, I, \delta)$ works as follows. For an input word $w \in \Sigma^*$, A starts in a nondeterministically chosen initial state $q_0 \in I$ with the word $w \cdot \lhd$ on its tape. This situation is described by the configuration $q_0 w \cdot \lhd$. Now assume that A is in a configuration of the form $xq_1w \cdot \lhd$, where $q_1 \in Q$ and $x, w \in \Sigma^*$, that is, A is in state q_1, the tape contains the word $xw \cdot \lhd$, and the head of A is on the first letter of the suffix $w \cdot \lhd$. Then A looks for the first occurrence from the left of a letter in w that is not translucent for state q_1, that is, if $w = uav$ such that $u \in (\tau(q_1))^*$, $v \in \Sigma^*$, and $a \in (\Sigma \setminus \tau(q_1))$, then A nondeterministically chooses a state $q_2 \in \delta(q_1, a)$, erases the letter a from the tape, thus producing the tape contents $xuv \cdot \lhd$, sets its internal state to q_2, and continues the computation from the configuration $xuq_2v \cdot \lhd$. In case $\delta(q_1, a) = \emptyset$, A halts without accepting. Finally, if $w \in (\tau(q_1))^*$, then A reaches the end-of-tape marker $\lhd$ and a transition from the set $\delta(q_1, \lhd)$ is applied. This transition is either an accept step or a state q_2 from Q. In the former case, A halts and accepts, while in the latter case, it continues the computation in state q_2 by reading its tape again from left to right, that is, from the configuration $q_2xw \cdot \lhd$. Finally, if $\delta(q_1, \lhd)$ is undefined, then A halts and rejects. Thus, the computation relation $\vdash_A$ that A induces on its set of configurations $\Sigma^* \cdot Q \cdot \Sigma^* \cdot \{\lhd\} \cup \{\mathsf{Accept}, \mathsf{Reject}\}$ is the reflexive and transitive closure $\vdash_A^*$ of the single-step computation relation $\vdash_A$ that is specified as follows, where x, u, v, w are words from Σ^* and a is a letter from Σ:

$$xqw \cdot \lhd \vdash_A \begin{cases} xuq'v \cdot \lhd, & \text{if } w = uav,\ u \in (\tau(q))^*,\ a \notin \tau(q),\ \text{and } q' \in \delta(q,a), \\ \mathsf{Reject}, & \text{if } w = uav,\ u \in (\tau(q))^*,\ a \notin \tau(q),\ \text{and } \delta(q,a) = \emptyset, \\ q'xw \cdot \lhd, & \text{if } w \in (\tau(q))^* \text{ and } q' \in \delta(q, \lhd), \\ \mathsf{Accept}, & \text{if } w \in (\tau(q))^* \text{ and } \delta(q, \lhd) = \mathsf{Accept}, \\ \mathsf{Reject}, & \text{if } w \in (\tau(q))^* \text{ and } \delta(q, \lhd) = \emptyset. \end{cases}$$

To describe computations of nrNFAwtls in a compact way, we introduce the notions of a sweep and a cycle.

Definition 2.6. Let $A = (Q, \Sigma, \lhd, \tau, I, \delta)$ be an nrNFAwtl.

(a) A *sweep* is a part of a computation of A in which the head moves from left to right across the complete tape contents. Thus, a sweep has the form $q_1wu \cdot \lhd \vdash_A^* w'q_2u \cdot \lhd$, where $q_1, q_2 \in Q$, $u \in (\tau(q_2))^*$, the word w' is obtained from w by deleting some letters or $w = w' = \lambda$ and $q_1 = q_2$, and the end-of-tape marker $\lhd$ is not visited during this partial computation. We use the notation

$$q_1wu \cdot \lhd \vdash_A^s w'uq_2 \cdot \lhd$$

to denote the above sweep. Observe that the configurations $w'q_2u \cdot \lhd$ and $w'uq_2 \cdot \lhd$ have exactly the same immediate successor configurations, as the word u only contains letters that are translucent for the state q_2.

(b) A *cycle* is a part of a computation of A that consists of a sweep $q_1wu \cdot \lhd \vdash_A^s w'uq_2 \cdot \lhd$ together with the next transitional step $q_3 \in \delta(q_2, \lhd)$. Thus, a cycle has the form $q_1wu \cdot \lhd \vdash_A^* w'q_2u \cdot \lhd \vdash_A q_3w'u \cdot \lhd$. We

use the notation

$$q_1wu \cdot \lhd \vdash_A^c q_3w'u \cdot \lhd$$

for this cycle.

A word $w \in \Sigma^*$ is accepted by the nrNFAwtl $A = (Q, \Sigma, \lhd, \tau, I, \delta)$ if there exists an initial state $q_0 \in I$ such that A has an accepting computation of the form $q_0w \cdot \lhd \vdash_A^* \mathsf{Accept}$. Then

$$L(A) = \{\, w \in \Sigma^* \mid w \text{ is accepted by } A \,\}$$

is the *language accepted by* A. We use $\mathcal{L}(\mathsf{nrNFAwtl})$ to denote the class of languages that are accepted by nrNFAwtls and $\mathcal{L}(\mathsf{nrDFAwtl})$ to denote the class of languages that are accepted by nrDFAwtls.

We illustrate these definitions by an example.

Example 2.7. Let $A = (Q, \{a, b, c\}, \lhd, \tau, \{q_a\}, \delta)$ be the nrNFAwtl that is defined by taking $Q = \{q_a, q_b, q_c, q_r\}$, $\tau(q_a) = \emptyset$, $\tau(q_b) = \{a\}$, $\tau(q_c) = \{b\}$, $\tau(q_r) = \{c\}$, and $\delta(q_a, a) = \{q_b\}$, $\delta(q_b, b) = \{q_c\}$, $\delta(q_c, c) = \{q_r\}$, $\delta(q_r, \lhd) = \{q_a\}$, $\delta(q_a, \lhd) = \mathsf{Accept}$. Given the word $w = aabbcc$ as input, the automaton A proceeds as follows:

$$\begin{array}{llllllll} q_aaabbcc \cdot \lhd & \vdash_A & q_babbcc \cdot \lhd & \vdash_A & aq_cbcc \cdot \lhd & \vdash_A & abq_rc \cdot \lhd \\ & \vdash_A & q_aabc \cdot \lhd & \vdash_A & q_bbc \cdot \lhd & \vdash_A & q_cc \cdot \lhd \\ & \vdash_A & q_r \cdot \lhd & \vdash_A & q_a \cdot \lhd & \vdash_A & \mathsf{Accept}, \end{array}$$

that is, A accepts on input $w = aabbcc$. In fact, $q_aaabbcc \cdot \lhd \vdash_A^s abcq_r \cdot \lhd$ is a sweep, and $q_aaabbcc \cdot \lhd \vdash_A^c q_aabc \cdot \lhd$ is a cycle of A. Actually, it is easily seen that $L(A) = \{\, a^nb^nc^n \mid n \geq 0 \,\}$. Furthermore, A is actually an nrDFAwtl.

Recall from [18] that the above language is not accepted by any NFAwtl. In fact, nrNFAwtls (or even nrDFAwtls) can also accept the following interesting languages:

$$\begin{array}{lllll} (1) & L^{(m)} & = & \{\, a_1^n a_2^n \cdots a_m^n \mid n \geq 0 \,\} \text{ for all } m \geq 1, \\ (2) & L_{\mathrm{cd}} & = & \{\, a^mb^nc^md^n \mid m, n \geq 0 \,\}, \text{ and} \\ (3) & L'_{\mathrm{copy}} & = & \{\, w\varphi(w) \mid w \in \{a, b\}^* \,\}, \end{array}$$

where φ is the morphism defined by $a \mapsto a'$ and $b \mapsto b'$ from Example 2.3.

As defined above, an nrNFAwtl $A = (Q, \Sigma, \lhd, \tau, I, \delta)$ may run into an infinite computation. Just assume that q is a state of A, $w \in (\tau(q))^*$, and $q \in \delta(q, \lhd)$. Then $qw \cdot \lhd \vdash_A qw \cdot \lhd \vdash_A qw \cdot \lhd$, and so forth. However, we can avoid this by converting A into an equivalent nrNFAwtl B as follows.

Let $B = (Q', \Sigma, \lhd, \tau', I', \delta')$, where $Q' = \{\, (q, S) \mid q \in Q \text{ and } S \subseteq Q \,\}$, $I' = \{\, (q, \emptyset) \mid q \in I \,\}$, $\tau'((q, S)) = \tau(q)$ for all $q \in Q$ and all $S \subseteq Q$,

$$\delta'((q, S), a) = \{\, (p, \emptyset) \mid p \in \delta(q, a) \,\}$$

for all $q \in Q$, $S \subseteq Q$, and all $a \in \Sigma$, and

$$\delta'((q, S), \lhd) = \{\, (p, S \cup \{q\}) \mid p \in \delta(q, \lhd) \text{ and } q \notin S \,\}$$

for all $q \in Q$ and all $S \subseteq Q$. Finally, take $\delta'((q, S), \lhd) = \mathsf{Accept}$ if $\delta(q, \lhd) = \mathsf{Accept}$. The set S is used to record those states in which the end-of-tape marker has been reached and the computation has continued. In the next cycle, when a non-translucent letter is read, then this set is emptied, otherwise, the next state is added to it.

This process continues until either a letter is read and deleted, or until no new state can be added to the current set S, in which case the computation fails. We illustrate this construction through a simple example.

Example 2.8. Let $A = (Q, \{a, b\}, \lhd, \tau, \{p\}, \delta)$, where $Q = \{p, q, r\}$, $\tau(p) = \tau(q) = \tau(r) = \{a\}$, and

$$\delta(p, b) = \{q\}, \delta(p, \lhd) = \{q, r\}, \delta(q, \lhd) = \{p\}, \delta(r, \lhd) = \mathsf{Accept},$$

and let $w = aabaa$. On input w, A can execute the following infinite computation:

$$pw \cdot \lhd = paabaa \cdot \lhd \vdash_A aaqaa \cdot \lhd \vdash_A paaaa \cdot \lhd \vdash_A qaaaa \cdot \lhd \vdash_A paaaa \cdot \lhd \vdash_A \cdots$$

The automaton $B = (Q', \{a, b\}, \lhd, \tau', \{p\}, \delta')$ that is obtained from A through the construction presented above simulates this computation as follows:

$$(p, \emptyset)aabaa \cdot \lhd \vdash_B aa(q, \emptyset)aa \cdot \lhd \vdash_B (p, \{q\})aaaa \cdot \lhd \vdash_B (q, \{p, q\})aaaa \cdot \lhd \vdash_B \mathsf{Reject},$$

that is, it recognizes the repetition and aborts the computation. Of course, using the transition $r \in \delta(p, \lhd)$ or $(r, \{p, q\}) \in \delta'((p, \{q\}), \lhd)$, both A and B can accept.

Moreover, an nrNFAwtl $A = (Q, \Sigma, \lhd, \tau, I, \delta)$ may accept without having read and deleted its input completely. This happens for the automata in Example 2.8, as $\tau(p) = \tau(r) = \{a\}$, $r \in \delta(p, \lhd)$, and $\delta(r, \lhd) = \mathsf{Accept}$. However, we can easily extend the nrNFAwtl A into an equivalent nrNFAwtl C that always reads and deletes its input completely before it accepts. Just take $C = (Q \cup \{q_e\}, \Sigma, \lhd, \tau', I, \delta')$, where q_e is a new state, $\tau'(q) = \tau(q)$ for all $q \in Q$ and $\tau'(q_e) = \emptyset$, and δ' is defined as follows:

$$\begin{array}{lrcl}
- & \delta'(q, a) & = & \delta(q, a) \quad \text{for all } q \in Q \text{ and all } a \in \Sigma, \\
- & \delta'(q, \lhd) & = & \begin{cases} \delta(q, \lhd), & \text{if } \delta(q, \lhd) \neq \mathsf{Accept}, \\ \{q_e\}, & \text{if } \delta(q, \lhd) = \mathsf{Accept}, \end{cases} \\
- & \delta'(q_e, a) & = & \{q_e\} \quad \text{for all } a \in \Sigma, \\
- & \delta'(q_e, \lhd) & = & \mathsf{Accept}.
\end{array}$$

Given a word $w \in \Sigma^*$ as input, the nrNFAwtl C will execute exactly the same steps as the nrNFAwtl A until A accepts. Now, the accept step of A is simulated by C through changing into state q_e. As $\tau'(q_e) = \emptyset$ and as $\delta'(q_e, a) = \{q_e\}$ for all $a \in \Sigma$, C will now read and delete the remaining tape contents and accept on reaching the end-of-tape marker $\lhd$. It easily follows that $L(C) = L(A)$. Together, the two constructions considered above yield the following technical result.

Lemma 2.9. *Each nrNFAwtl A can be effectively converted into an equivalent nrNFAwtl C that never gets into an infinite computation and that accepts only after reading and deleting its input completely. In addition, if A is deterministic, then so is C.*

Hence, in what follows, we can always assume that an nrNFAwtl or an nrDFAwtl considered has the additional properties guaranteed by this lemma.

3. Some inclusion results and closure properties

Next, we show that the nrNFAwtl is indeed an extension of the NFAwtl.

Theorem 3.1. *From a given NFAwtl A, one can effectively construct an nrNFAwtl B such that $L(B) = L(A)$. In addition, if A is deterministic, then so is B.*

Proof. Let $A = (Q, \Sigma, \lhd, \tau, I, F, \delta)$ be an NFAwtl. We define an nrNFAwtl $B = (Q_B, \Sigma, \lhd, \tau_B, I_B, \delta_B)$ that simulates the computations of A as follows:

- $Q_B = Q \cup \{ q' \mid q \in Q \}$, where for each state $q \in Q$, q' is an additional auxiliary state, and $I_B = I$,
- for each state $q \in Q$, $\tau_B(q) = \tau(q)$ and $\tau_B(q') = \Sigma$,
- for each state $q \in Q$ and each letter $a \in \Sigma$, $\delta_B(q,a) = \{ p' \mid p \in \delta(q,a) \}$ and $\delta_B(q',a) = \emptyset$.
- Furthermore, for each state $q \in Q$, $\delta_B(q, \lhd) = \mathsf{Accept}$, if $q \in F$, and $\delta_B(q, \lhd) = \emptyset$, otherwise. Finally, $\delta_B(q', \lhd) = \{q\}$.

It remains to verify that B just simulates the computations of A.

Assume that $qw \cdot \lhd$ is a configuration of A, that is, $q \in Q$ and $w \in \Sigma^*$. From the definition of the computation relation $\vdash_A$, we see that there are four different cases that we must consider.

First, if $w = uav$ for some words $u \in (\tau(q))^*$, $v \in \Sigma^*$, and a letter $a \in (\Sigma \setminus \tau(q))$, then A executes a transition from $\delta(q,a)$.

- If $p \in \delta(q,a)$, then $qw \cdot \lhd = quav \cdot \lhd \vdash_A puv \cdot \lhd$ is a possible step of A. In this case, B can execute the following sequence of steps:

$$qw \cdot \lhd = quav \cdot \lhd \vdash_B up'v \cdot \lhd \vdash_B puv \cdot \lhd.$$

- On the other hand, if $\delta(q,a) = \emptyset$, then A halts and rejects. However, in this case, also $\delta_B(q,a) = \emptyset$, and hence, B halts and rejects as well.

Secondly, if $w \in (\tau(q))^*$, then A halts.

- If $q \in F$, then A accepts. In this case, $\delta_B(q, \lhd) = \mathsf{Accept}$, that is, B halts and accepts, too.
- If $q \notin F$, then A rejects. In this case, $\delta_B(q, \lhd) = \emptyset$, that is, B halts and rejects as well.

It follows that $L(A) \subseteq L(B)$.

Conversely, if $w \in L(B)$, then it is easily verified that each accepting computation of B on input w is just a simulation of an accepting computation of A on input w. Thus, we see that $L(B) = L(A)$.

Finally, the above definition of B shows that B is deterministic, if A is. This completes the proof of Theorem 3.1. □

Together with Example 2.7, this theorem has the following consequence.

Corollary 3.2. $\mathcal{L}(\mathsf{NFAwtl}) \subsetneq \mathcal{L}(\mathsf{nrNFAwtl})$ *and* $\mathcal{L}(\mathsf{DFAwtl}) \subsetneq \mathcal{L}(\mathsf{nrDFAwtl})$.

It is known that all languages accepted by NFAwtls are necessarily semi-linear, that is, their images with respect to the Parikh mapping are semi-linear subsets of $\mathbb{N}^m$, where $\mathbb{N}$ is the set of nonnegative integers and m is the cardinality of the alphabet used. Does a corresponding result also hold for nrNFAwtls? First, we consider this question for the special case of a unary alphabet.

Proposition 3.3. *A language $L \subseteq \{a\}^*$ is accepted by an nrNFAwtl if and only if it is a regular language.*

Proof. If $L \subseteq \{a\}^*$ is a regular language, then it is obviously accepted by an nrNFAwtl.

Conversely, assume that an nrNFAwtl $A = (Q, \{a\}, \lhd, \tau, I, \delta)$ accepts a language $L \subseteq \{a\}^*$. By Lemma 2.9, we can assume that the nrNFAwtl A never gets into an infinite computation and that it accepts only after reading and deleting its input completely. From A, we now construct an NFA with λ-transitions $B = (Q, \{a\}, I, F, \delta_B)$ by taking $F = \{ q \in Q \mid \delta(q, \lhd) = \mathsf{Accept} \}$ and by defining the transition relation δ_B as follows:

$$\begin{array}{lllll} (1) & \delta_B(q,a) & = & \delta(q,a) & \text{for all } q \in Q, \\ (2) & \delta_B(q,\lambda) & = & \delta(q,\lhd), & \text{if } \delta(q,\lhd) \subseteq Q. \end{array}$$

We claim that $L(B) = L(A) = L$ holds, which then implies that L is a regular language.

For each state $q \in Q$, if $\tau(q) \neq \emptyset$, then $\tau(q) = \{a\}$ and $\delta(q, a) = \emptyset$. Hence, for all $m \geq 1$,

$$qa^m \cdot \lhd \vdash_A \begin{cases} q'a^m \cdot \lhd, & \text{if } q' \in \delta(q, \lhd), \\ \mathsf{Accept}, & \text{if } \delta(q, \lhd) = \mathsf{Accept}, \\ \mathsf{Reject}, & \text{if } \delta(q, \lhd) = \emptyset. \end{cases}$$

On the other hand, for each state $q \in Q$ for which $\tau(q) = \emptyset$,

$$qa^m \cdot \lhd \vdash_A \begin{cases} q'a^{m-1} \cdot \lhd, & \text{if } q' \in \delta(q, a), \\ \mathsf{Reject}, & \text{if } \delta(q, a) = \emptyset. \end{cases}$$

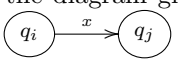
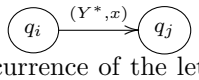
Hence, if a^m is accepted by the nrNFAwtl A, then a corresponding accepting computation of A reads (and deletes) the word a^m simply letter by letter from left to right, where this sequence of computational steps may be interspersed with steps that change the state without reading (and deleting) an occurrence of the letter a. Now, it is easily seen that the NFA B can execute the very same computation. Conversely, each accepting computation of the NFA B just mirrors an accepting computation of the nrNFAwtl A. This completes the proof of Proposition 3.3. □

Thus, all unary languages that are accepted by nrNFAwtls are semi-linear. For non-unary alphabets, the corresponding question is still open. To illustrate this problem, we consider the following detailed example.

Example 3.4. We define the nrDFAwtl $A_{\text{ex3}} = (Q, \Sigma, \lhd, \tau, q_0, \delta)$ as follows:

- $Q = \{q_0, q_1, q_2, q_3, q_4, q_5, q_6, q_7, q_8\}$ and $\Sigma = \{a, b, c\}$,
- $\begin{array}{llllllll} \tau(q_0) & = & \{a\}, & \tau(q_1) & = & \tau(q_2) & = & \emptyset, \\ \tau(q_3) & = & \{b\}, & \tau(q_4) & = & \tau(q_5) & = & \emptyset, \\ \tau(q_6) & = & \{a, c\}, & \tau(q_7) & = & \tau(q_8) & = & \emptyset, \end{array}$
- and the transition function δ is defined through

$$\begin{array}{llll} (1)\ \delta(q_0, b) = q_1, & (4)\ \delta(q_2, \lhd) = q_7, & (7)\ \delta(q_5, b) = q_6, & (10)\ \delta(q_7, a) = q_8, \\ (2)\ \delta(q_1, c) = q_2, & (5)\ \delta(q_3, c) = q_4, & (8)\ \delta(q_6, b) = q_1, & (11)\ \delta(q_8, \lhd) = \mathsf{Accept}. \\ (3)\ \delta(q_2, a) = q_3, & (6)\ \delta(q_4, a) = q_5, & (9)\ \delta(q_6, \lhd) = q_0, & \end{array}$$

We can actually describe the nrDFAwtl A_{ex3} through the diagram given in Figure 1. In this diagram, the vertices correspond to the states of A, an edge of the form  denotes a transition from q_i to q_j that simply reads an occurrence of the letter x, and an edge of the form  denotes a transition from q_i to q_j in which a factor from Y^* is skipped and a subsequent occurrence of the letter x is read. Finally, an edge, the label of which contains the end-of-tape marker $\lhd$, corresponds to starting a new sweep or an accept operation.

From this diagram, we can easily extract the following information on the computations of A_{ex3}:

1. The shortest path from q_0 to Accept removes a single occurrence of each of the letters a, b, and c.
2. A sweep of the automaton A starts in q_0 and ends in q_2 or in q_6, or it starts in q_7.
3. A sweep may contain one or more repetitions of the cycle $q_6 \to q_1 \to q_2 \to q_3 \to q_4 \to q_5 \to q_6$, during which two occurrences of each of the letters a, b, and c are removed.
4. During a sweep that ends at q_6, the same even number of occurrences of each of the letters a, b, and c are removed.
5. The last part of an accepting computation leads from q_0 or from q_6 to q_1, then to q_2, then to q_7, and finally to q_8. Thus, during this part, a single occurrence of each of the letters a, b, and c is removed.

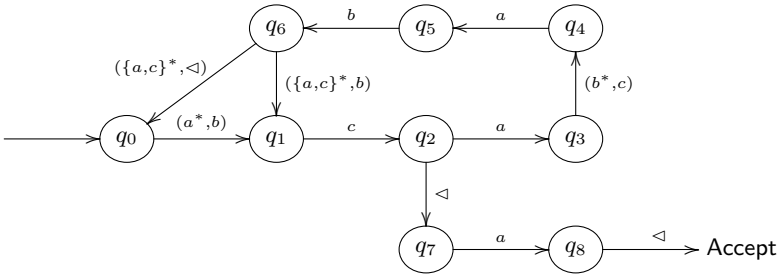


FIGURE 1. The diagram describing the nrDFAwtl A_{ex3}.

Together these observations imply that during each accepting computation, A_{ex3} removes the same odd number of occurrences of the letters a, b, and c. This implies that the Parikh image $\pi(L(A_{\text{ex3}}))$ of the language $L(A_{\text{ex3}})$ satisfies the inclusion $\pi(L(A_{\text{ex3}})) \subseteq \{\, (2n+1, 2n+1, 2n+1) \mid n \geq 0 \,\}$.

We now consider an input of the form $(abc)^{3n}$ for some $n \geq 2$. We shall see that $q_0(abc)^{3n} \cdot \lhd \vdash^*_{A_{\text{ex3}}} q_0(abc)^n \cdot \lhd$ holds. To prove this statement, we first establish the following technical results.

Claim 1.
(a) $q_0(abc)^n \cdot \lhd \vdash^*_{A_{\text{ex3}}} aq_2(abc)^{n-1} \cdot \lhd$ for all $n \geq 1$.
(b) $(abc)^k aq_2(abc)^m \cdot \lhd \vdash^*_{A_{\text{ex3}}} (abc)^{k+1} aq_2(abc)^{m-3} \cdot \lhd$ for all $k \geq 0$ and $m \geq 3$.
(c) $(abc)^{n-1} aq_2(abc)^2 \cdot \lhd \vdash^*_{A_{\text{ex3}}} q_0(abc)^n \cdot \lhd$ for all $n \geq 1$.

Proof. (a) For $n \geq 1$, we obtain the following initial part of a computation of A_{ex3} on input $(abc)^n$:

$$q_0(abc)^n \cdot \lhd = q_0abc(abc)^{n-1} \cdot \lhd \vdash_{A_{\text{ex3}}} aq_1c(abc)^{n-1} \cdot \lhd \vdash_{A_{\text{ex3}}} aq_2(abc)^{n-1} \cdot \lhd,$$

which proves the statement in (a).

(b) For all $k \geq 0$ and $m \geq 3$, we have the following partial computation of A_{ex3}:

$$\begin{aligned}
(abc)^k aq_2(abc)^m \cdot \lhd &= (abc)^k aq_2abc(abc)^{m-1} \cdot \lhd \\
&\vdash_{A_{\text{ex3}}} (abc)^k aq_3bc(abc)^{m-1} \cdot \lhd \\
&\vdash_{A_{\text{ex3}}} (abc)^k abq_4abc(abc)^{m-2} \cdot \lhd \\
&\vdash_{A_{\text{ex3}}} (abc)^k abq_5bc(abc)^{m-2} \cdot \lhd \\
&\vdash_{A_{\text{ex3}}} (abc)^k abq_6cabc(abc)^{m-3} \cdot \lhd \\
&\vdash_{A_{\text{ex3}}} (abc)^{k+1} aq_1c(abc)^{m-3} \cdot \lhd \\
&\vdash_{A_{\text{ex3}}} (abc)^{k+1} aq_2(abc)^{m-3} \cdot \lhd,
\end{aligned}$$

which proves (b).

(c) Finally, for $n \geq 1$, we have the following partial computation of A_{ex3}:

$$\begin{aligned}
(abc)^{n-1} aq_2(abc)^2 \cdot \lhd &= (abc)^{n-1} aq_2abcabc \cdot \lhd \vdash_{A_{\text{ex3}}} (abc)^{n-1} aq_3bcabc \cdot \lhd \\
&\vdash_{A_{\text{ex3}}} (abc)^{n-1} abq_4abc \cdot \lhd \vdash_{A_{\text{ex3}}} (abc)^{n-1} abq_5bc \cdot \lhd \\
&\vdash_{A_{\text{ex3}}} (abc)^{n-1} abq_6c \cdot \lhd \vdash_{A_{\text{ex3}}} q_0(abc)^n \cdot \lhd,
\end{aligned}$$

which proves (c). □

By putting these partial computations together, we obtain the following computation for all $n \geq 2$ and all $1 \leq k \leq n-1$:

$$\begin{aligned} q_0(abc)^{3n} \cdot \lhd &\vdash^*_{A_{\mathrm{ex3}}} aq_2(abc)^{3n-1} \cdot \lhd \\ &\vdash^*_{A_{\mathrm{ex3}}} (abc)aq_2(abc)^{3n-1-3} \cdot \lhd \quad \vdash^*_{A_{\mathrm{ex3}}} (abc)^2aq_2(abc)^{3n-1-6} \cdot \lhd \\ &\vdash^*_{A_{\mathrm{ex3}}} (abc)^kaq_2(abc)^{3n-1-3k} \cdot \lhd \vdash^*_{A_{\mathrm{ex3}}} (abc)^{n-1}aq_2(abc)^2 \cdot \lhd \\ &\vdash^*_{A_{\mathrm{ex3}}} q_0(abc)^n \cdot \lhd. \end{aligned}$$

In addition,

$$q_0abc \cdot \lhd \vdash_{A_{\mathrm{ex3}}} aq_1c \cdot \lhd \vdash_{A_{\mathrm{ex3}}} aq_2 \cdot \lhd \vdash_{A_{\mathrm{ex3}}} q_7a \cdot \lhd \vdash_{A_{\mathrm{ex3}}} q_8 \cdot \lhd \vdash_{A_{\mathrm{ex3}}} \mathsf{Accept}.$$

Hence, it follows that $L_{\mathrm{ex3}} = \{\, (abc)^{3^n} \mid n \geq 0 \,\} \subseteq L(A_{\mathrm{ex3}})$. Unfortunately, A_{ex3} also accepts some words that do not belong to the language L_{ex3}. In fact, it can be shown that $L' = \{\, ab(cacabb)^nc \mid n \geq 0 \,\} \subseteq L(A_{\mathrm{ex3}})$. Indeed, for $n = 0$, we have $ab(cacabb)^nc = abc \in L(A_{\mathrm{ex3}})$. Now, proceeding by induction on n,

$$\begin{aligned} q_0ab(cacabb)^{n+1}c \cdot \lhd \quad &= \quad q_0abcacabb(cacabb)^nc \cdot \lhd \vdash_{A_{\mathrm{ex3}}} aq_1cacabb(cacabb)^nc \cdot \lhd \\ &\vdash^6_{A_{\mathrm{ex3}}} aq_1(cacabb)^nc \cdot \lhd \qquad \vdash^*_{A_{\mathrm{ex3}}} \mathsf{Accept}. \end{aligned}$$

As $\pi(ab(cacabb)^nc) = (2n+1, 2n+1, 2n+1)$, we see that

$$\pi(L(A_{\mathrm{ex3}})) = \{\, (2n+1, 2n+1, 2n+1) \mid n \geq 0 \,\},$$

which shows that the language $L(A_{\mathrm{ex3}})$ is in fact semi-linear.

On the other hand, we have the following fact.

Claim 2. $L(A_{\mathrm{ex3}}) \cap \{abc\}^* = L_{\mathrm{ex3}}$.

Proof. If $w = (abc)^{3n+1}$, then

$$q_0w \cdot \lhd \quad = \quad q_0(abc)^{3n+1} \cdot \lhd \quad \vdash^*_{A_{\mathrm{ex3}}} \quad (abc)^naq_2 \cdot \lhd \quad \vdash_{A_{\mathrm{ex3}}} \quad q_7(abc)^na \cdot \lhd,$$

and from the configuration $q_7(abc)^na \cdot \lhd$, A_{ex3} accepts only if $n = 0$. Analogously, if $w = (abc)^{3n+2}$, then

$$\begin{aligned} q_0w \cdot \lhd \quad &= \quad q_0(abc)^{3n+2} \cdot \lhd \vdash^*_{A_{\mathrm{ex3}}} (abc)^naq_2abc \cdot \lhd \vdash_{A_{\mathrm{ex3}}} (abc)^naq_3bc \cdot \lhd \\ &\vdash_{A_{\mathrm{ex3}}} (abc)^nabq_4 \cdot \lhd \quad \vdash_{A_{\mathrm{ex3}}} \mathsf{Reject}. \end{aligned}$$

Therefore, the only powers of abc that A_{ex3} accepts are those of the form $(abc)^m$ for which m is a power of three. □

Our example shows that the intersection of a language that is accepted by an nrDFAwtl with a regular set is not necessarily semi-linear.

At this point, it remains open whether the class $\mathcal{L}(\mathsf{nrDFAwtl})$ contains any non-unary languages that are not semi-linear.

As all rational trace languages are accepted by NFAwtls, Corollary 3.2 implies that all rational trace languages are accepted by nrNFAwtls. However, as shown in [21], the rational trace language

$$L_\vee = \{\, w \in \{a,b\}^* \mid \exists n \geq 0 : |w|_a = n \text{ and } |w|_b \in \{n, 2n\} \,\}$$

is not accepted by any DFAwtl. Our next result shows that this language is not even accepted by any nrDFAwtl.

Proposition 3.5. $L_\vee \notin \mathcal{L}(\mathsf{nrDFAwtl})$.

Proof. Assume that $A = (Q, \Sigma, \lhd, \tau, I, \delta)$ is an nrDFAwtl that accepts the language $L_\vee$, where $Q = \{q_0, q_1, \ldots, q_{m-1}\}$, $\Sigma = \{a, b\}$, and $I = \{q_0\}$.

Claim. If $q_i a^r b^s \cdot \lhd \vdash_A^s a^{r-r_1} b^{s-s_1} q_j \cdot \lhd$ is a sweep within an accepting computation of A on input $a^n b^n$ or $a^n b^{2n}$, then $r_1 \leq m$ and $s_1 \leq m$.

Proof. As $L(A) = L_\vee$, and as an nrDFAwtl only deletes letters during its computation, we see that $w = a^r b^s$ is converted into $a^{r-r_1} b^{s-s_1}$ for some $0 \leq r_1 \leq r$ and $0 \leq s_1 \leq s$. Thus, during the above sweep, A first reads (and deletes) r_1 occurrences of the letter a, and then it reads (and deletes) s_1 occurrences of the letter b. If $r_1 > m$, then some state of A appears at least twice while the head of A is still inside the prefix a^r. This implies that by using pumping, A can also execute the sweeps of the form

$$q_i a^{r+\mu \cdot t} b^s \cdot \lhd \vdash_A^s a^{r-r_1} b^{s-s_1} q_j \cdot \lhd$$

for all $\mu \geq 1$ and some value $1 \leq t \leq m$. But then, together with $a^n b^n$ or $a^n b^{2n}$, A would also accept the words $a^{n+\mu \cdot t} b^n$ or $a^{n+\mu \cdot t} b^{2n}$, a contradiction. It follows that $r_1 \leq m$, and analogously, it can be shown that $s_1 \leq m$. □

Let $n > 3m^2$. By Lemma 2.9, we can assume that A reads and deletes its input completely before it accepts. As $a^n b^n \in L_\vee$, the computation of A on input $a^n b^n$ is accepting. It consists of a sequence of sweeps and an accept step, that is, we have

$$\begin{array}{lcccl} q_0 a^n b^n \cdot \lhd & \vdash_A^s & a^{n-r_1} b^{n-s_1} q_{i_1} \cdot \lhd & \vdash_A & q_{j_1} a^{n-r_1} b^{n-s_1} \cdot \lhd \\ & \vdash_A^s & a^{n-r_1-r_2} b^{n-s_1-s_2} q_{i_2} \cdot \lhd & \vdash_A & q_{j_2} a^{n-r_1-r_2} b^{n-s_1-s_2} \cdot \lhd \\ & \vdash_A^s & \cdots & \vdash_A^s & a^{n-r_1-r_2-\cdots-r_k} b^{n-s_1-s_2-\cdots-s_k} q_{i_k} \cdot \lhd \\ & \vdash_A & \mathsf{Accept}, & & \end{array}$$

where $k \geq 1$ and $r_i, s_i \leq m$ for all $i = 1, 2, \ldots, k$. Furthermore, since A accepts only after completely deleting its input, we have $n = r_1 + r_2 + \cdots + r_k = s_1 + s_2 + \cdots + s_k$, which implies that $k > 3m$.

As A has only m states, it follows that there are indices $1 \leq \alpha < \beta \leq m+1$ such that the states q_{j_α} and q_{j_β} are identical. Therefore, the above computation can be written as follows:

$$\begin{array}{lcl} q_0 a^n b^n \cdot \lhd & \vdash_A^* & q_{j_\alpha} a^{n-r_1-r_2-\cdots-r_\alpha} b^{n-s_1-s_2-\cdots-s_\alpha} \cdot \lhd \\ & \vdash_A^* & q_{j_\beta} a^{n-r_1-r_2-\cdots-r_\alpha-r_{\alpha+1}-\cdots-r_\beta} b^{n-s_1-s_2-\cdots-s_\alpha-s_{\alpha+1}-\cdots-s_\beta} \cdot \lhd \\ & = & q_{j_\alpha} a^{n-r_1-r_2-\cdots-r_\alpha-r_{\alpha+1}-\cdots-r_\beta} b^{n-s_1-s_2-\cdots-s_\alpha-s_{\alpha+1}-\cdots-s_\beta} \cdot \lhd \\ & \vdash_A^* & q_{i_k} \cdot \lhd \vdash_A \mathsf{Accept}. \end{array}$$

To simplify the notation, we take $n_\alpha = n - r_1 - r_2 - \cdots - r_\alpha$, $c = r_{\alpha+1} + \cdots + r_\beta$, $n'_\alpha = n - s_1 - s_2 - \cdots - s_\alpha$, and $c' = s_{\alpha+1} + \cdots + s_\beta$. Then we also have the following accepting computation:

$$\begin{array}{lcl} q_0 a^{n+c} b^{n+c'} \cdot \lhd & \vdash_A^* & q_{j_\alpha} a^{n_\alpha+c} b^{n'_\alpha+c'} \cdot \lhd \\ & \vdash_A^* & q_{j_\alpha} a^{n_\alpha-c+c} b^{n'_\alpha-c'+c'} \cdot \lhd \\ & = & q_{j_\alpha} a^{n_\alpha} b^{n'_\alpha} \cdot \lhd \\ & \vdash_A^* & \mathsf{Accept}. \end{array}$$

Thus, $a^{n+c} b^{n+c'} \in L_\vee$. As $c \leq m^2$ and $c' \leq m^2$, while $n > 3m^2$, it follows that $n + c = n + c'$, which in turn implies that $c = c'$.

Now we consider the accepting computation of A for the input $a^n b^{2n} \in L_\vee$. As A is deterministic, this computation looks as follows:

$$\begin{array}{rcl} q_0 a^n b^{2n} \cdot \lhd & \vdash_A^* & q_{j_\alpha} a^{n_\alpha} b^{n+n'_\alpha} \cdot \lhd \\ & \vdash_A^* & q_{j_\alpha} a^{n_\alpha - c} b^{n+n'_\alpha - c} \cdot \lhd \\ & \vdash_A^* & \mathsf{Accept}. \end{array}$$

However, A can then also execute the following accepting computation:

$$\begin{array}{rcl} q_0 a^{n+c} b^{2n+c} \cdot \lhd & \vdash_A^* & q_{j_\alpha} a^{n_\alpha + c} b^{n+n'_\alpha + c} \cdot \lhd \\ & \vdash_A^* & q_{j_\alpha} a^{n_\alpha - c + c} b^{n+n'_\alpha - c + c} \cdot \lhd \\ & = & q_{j_\alpha} a^{n_\alpha} b^{n+n'_\alpha} \cdot \lhd \\ & \vdash_A^* & \mathsf{Accept}. \end{array}$$

Thus, $a^{n+c} b^{2n+c} \in L_\vee$. However, $n + c < 2n + c < 2(n + c)$, which shows that $a^{n+c} b^{2n+c} \notin L_\vee$, a contradiction. Hence, it follows that $L_\vee$ is not accepted by any nrDFAwtl. □

As $L_\vee$ is a rational trace language, the above result implies that the class of rational trace languages is not even contained in the language class $\mathcal{L}(\mathsf{nrDFAwtl})$.

Finally, we present some closure and non-closure properties for the classes of languages that are accepted by nrNFAwtls and by nrDFAwtls.

Theorem 3.6. *The language class $\mathcal{L}(\mathsf{nrNFAwtl})$ is closed under union and disjoint shuffle.*

Proof. Let $A_1 = (Q_1, \Sigma, \lhd, \tau_1, I_1, \delta_1)$ and $A_2 = (Q_2, \Sigma, \lhd, \tau_2, I_2, \delta_2)$ be nrNFAwtls. Without loss of generality, we may assume that the sets of states Q_1 and Q_2 are disjoint. Let $A = (Q_1 \cup Q_2, \Sigma, \lhd, \tau, I_1 \cup I_2, \delta)$ be the nrNFAwtl that is defined by taking

$$\tau(q) = \begin{cases} \tau_1(q), & \text{if } q \in Q_1, \\ \tau_2(q), & \text{if } q \in Q_2, \end{cases}$$

and, for all $a \in \Sigma \cup \{\lhd\}$,

$$\delta(q, a) = \begin{cases} \delta_1(q, a), & \text{if } q \in Q_1, \\ \delta_2(q, a), & \text{if } q \in Q_2. \end{cases}$$

Then $L(A) = L(A_1) \cup L(A_2)$, which proves that the class $\mathcal{L}(\mathsf{nrNFAwtl})$ is closed under union.

Let $L_1 \subseteq \Sigma_1^*$ and $L_2 \subseteq \Sigma_2^*$, where the alphabets Σ_1 and Σ_2 are disjoint. If $A_1 = (Q_1, \Sigma_1, \lhd, \tau_1, I_1, \delta_1)$ and $A_2 = (Q_2, \Sigma_2, \lhd, \tau_2, I_2, \delta_2)$ are nrNFAwtls with disjoint sets of states such that $L(A_1) = L_1$ and $L(A_2) = L_2$, then we obtain an nrNFAwtl $A = (Q_1 \cup Q_2, \Sigma_1 \cup \Sigma_2, \lhd, \tau, I_1, \delta)$ for the shuffle language $\mathrm{sh}(L_1, L_2)$ by taking

$$\begin{array}{rcl} \tau(q) & = & \begin{cases} \tau_1(q) \cup \Sigma_2, & \text{if } q \in Q_1, \\ \tau_2(q) \cup \Sigma_1, & \text{if } q \in Q_2, \end{cases} \\ \delta(q, a) & = & \begin{cases} \delta_1(q, a), & \text{if } q \in Q_1 \text{ and } a \in \Sigma_1, \\ \delta_2(q, a), & \text{if } q \in Q_2 \text{ and } a \in \Sigma_2, \end{cases} \end{array}$$

and

$$\delta(q,\lhd) \;=\; \begin{cases} \delta_1(q,\lhd), & \text{if } q \in Q_1 \text{ and } \delta_1(q,\lhd) \subseteq Q_1,\\ I_2, & \text{if } q \in Q_1 \text{ and } \delta_1(q,\lhd) = \mathsf{Accept},\\ \delta_2(q,\lhd), & \text{if } q \in Q_2. \end{cases}$$

Given a word $w \in (\Sigma_1 \cup \Sigma_2)^*$ as input, A starts in a state from I_1, and it behaves just like the automaton A_1, ignoring all letters from Σ_2. If and when the end-of-tape marker $\lhd$ is reached in a state $q \in Q_1$ for which $\delta_1(q,\lhd) = \mathsf{Accept}$ holds, then A enters a state from the set I_2 and continues its computation by simulating A_2, this time ignoring all letters from Σ_1 that may still be on its tape. Finally, A accepts if and when the computation of A_2 accepts. It follows that $L(A) = \mathrm{sh}(L(A_1), L(A_2)) = \mathrm{sh}(L_1, L_2)$. Thus, the class $\mathcal{L}(\mathsf{nrNFAwtl})$ is closed under disjoint shuffle. □

For nrDFAwtls, we have the following results.

Theorem 3.7. *The language class $\mathcal{L}(\mathsf{nrDFAwtl})$ is closed under complementation and disjoint shuffle, but it is neither closed under union nor under intersection. Moreover, this class is not closed under alphabetic morphisms.*

Proof. The language $L_1 = \{\, w \in \{a,b\}^* \mid |w|_a = |w|_b \,\}$ and the language $L_2 = \{\, w \in \{a,b\}^* \mid |w|_b = 2 \cdot |w|_a \,\}$ are accepted by DFAwtls. However, $L_1 \cup L_2 = L_\vee$, which is not even accepted by any nrDFAwtl by Proposition 3.5. This shows that the class $\mathcal{L}(\mathsf{nrDFAwtl})$ is not closed under union.

Next, we prove that the class $\mathcal{L}(\mathsf{nrDFAwtl})$ is closed under complementation. Let $A = (Q, \Sigma, \lhd, \tau, I, \delta)$ be an nrDFAwtl such that $L(A) = L \subseteq \Sigma^*$. Recall from Lemma 2.9 that we can assume that all computations of A are finite. We define an nrDFAwtl $A^c = (Q \cup \{q_+\}, \Sigma, \lhd, \tau^c, I, \delta^c)$, where q_+ is a new state, by taking

$$\tau^c(q) \;=\; \begin{cases} \tau(q), & \text{if } q \in Q,\\ \Sigma, & \text{if } q = q_+, \end{cases}$$

and

$$\begin{aligned} \delta^c(q,a) \;&=\; \begin{cases} \delta(q,a), & \text{if } \delta(q,a) \in Q,\\ q_+, & \text{if } a \notin \tau(q) \text{ and } \delta(q,a) \text{ is undefined},\\ \emptyset, & \text{if } a = \lhd \text{ and } \delta(q,\lhd) = \mathsf{Accept}, \end{cases}\\ \delta^c(q_+,\lhd) \;&=\; \mathsf{Accept}. \end{aligned}$$

Given a word $w \in \Sigma^*$ as input, the automaton A^c simulates the computation of the automaton A on input w step by step until A either accepts or gets stuck. In the former case, A^c reaches the end-of-tape marker $\lhd$ and gets stuck, while in the latter case, it enters the state q_+ and accepts. It follows that $L(A^c) = \Sigma^* \setminus L(A)$, which shows that the class $\mathcal{L}(\mathsf{nrDFAwtl})$ is closed under complementation.

Closure under complementation and non-closure under union imply that the class $\mathcal{L}(\mathsf{nrDFAwtl})$ is not closed under intersection. Furthermore, closure under disjoint shuffle is proved in the same way as for nrNFAwtls.

Finally, let $\Sigma = \{a,b,c\}$, let

$$L = \{\, w \in \{a,b\}^* \mid |w|_a = |w|_b \,\} \cup \{\, w \in \{a,c\}^* \mid |w|_c = 2 \cdot |w|_a \,\},$$

and let $\varphi : \Sigma^* \to \{a,b\}^*$ be the alphabetic morphism that is defined through $a \mapsto a$, $b \mapsto b$, and $c \mapsto b$. It is easily verified that the language L is accepted by a DFAwtl. However, $\varphi(L) = L_\vee$, which is not accepted by any nrDFAwtl by Proposition 3.5. This proves that the class $\mathcal{L}(\mathsf{nrDFAwtl})$ is not closed under alphabetic morphisms. □

As we shall see later, using another example language (see Cor. 4.20), the class $\mathcal{L}(\mathsf{nrNFAwtl})$ is not closed under alphabetic morphisms, either. However, many questions concerning closure and non-closure are still open for non-returning finite automata with translucent letters.

4. Comparing finite automata with translucent letters to various types of jumping finite automata

As it will serve as a reference, we restate, in short, the definition of the finite automaton.

Definition 4.1. A *nondeterministic finite automaton*, or an *NFA*, is given through a 5-tuple $A = (Q, \Sigma, I, F, \delta)$, where Q is a finite set of states, Σ is a finite alphabet, $I, F \subseteq Q$ are the sets of initial and final states, respectively, and $\delta : Q \times \Sigma \to \mathcal{P}(Q)$ is a transition relation. The NFA A is *deterministic*, or a *DFA*, if $|I| = 1$ and $|\delta(q, a)| \leq 1$ for all pairs $(q, a) \in Q \times \Sigma$.

A *configuration* of an NFA A is a word qw from the set $Q \cdot \Sigma^*$. Here, q is the current state, and w is the still unread part of the input. A configuration of the form $q_0 w$ is called an *initial configuration*, if $q_0 \in I$. The *computation relation* $\vdash_A^*$ that A induces on its set of configurations is the reflexive and transitive closure of the single-step computation relation $\vdash_A$ that is defined as follows:

$$qw \vdash q'w' \text{ if } w = aw' \text{ for some } a \in \Sigma, w' \in \Sigma^*, \text{ and } q' \in \delta(q, a).$$

Then $L(A) = \{ w \in \Sigma^* \mid \exists q_0 \in I \, \exists q_f \in F : q_0 w \vdash_A^* q_f \}$ is the language accepted by the NFA A.

Observe that an NFA A reads its input strictly from left to right, letter by letter. It is well-known that the language class $\mathcal{L}(\mathsf{NFA})$ of languages that are accepted by NFAs is the class REG of regular languages, and that this class coincides with the class $\mathcal{L}(\mathsf{DFA})$ of languages that are accepted by DFAs. Next, we recall the definition of the jumping finite automaton from [11].

Definition 4.2. A *jumping finite automaton*, or a *JFA*, is given through a 5-tuple $J = (Q, \Sigma, I, F, \delta)$, where all components are defined as for an NFA. However, the computation relation of J is defined very differently.

A *configuration* of J is any word uqv from the set $\Sigma^* \cdot Q \cdot \Sigma^*$. The *computation relation* $\curvearrowright_J^*$ that J induces on its set of configurations is the reflexive and transitive closure of the single-step *jumping relation* $\curvearrowright_J$ that is defined as follows, where $q, q' \in Q$, $x, z, x', z' \in \Sigma^*$, and $a \in \Sigma$:

$$xqaz \curvearrowright_J x'q'z' \text{ if } xz = x'z' \text{ and } q' \in \delta(q, a),$$

that is, when reading the letter a in state q, then J consumes that letter, changes to state q', and moves its head to an arbitrary letter of the remaining tape inscription. Consequently,

$$L(J) = \{ uv \mid u, v \in \Sigma^* \text{ and } \exists q_0 \in I \, \exists q_f \in F : uq_0 v \curvearrowright_J^* q_f \}$$

is the language accepted by the JFA J.

It has been observed that the JFA accepts some languages that are not even context-free, *e.g.*, the language

$$L_{eq3} = \{ w \in \{a, b, c\}^* \mid |w|_a = |w|_b = |w|_c \},$$

while, on the other hand, it does not even accept the finite language $L_f = \{ab\}$. Thus, the class $\mathcal{L}(\mathsf{JFA})$ of languages that are accepted by jumping finite automata is incomparable under inclusion to the finite languages, the regular languages, and the (deterministic) context-free languages. In fact, the following characterization has been established for this language class.

Theorem 4.3. [5] *A language L is accepted by a JFA if and only if L is commutative and semi-linear.*

Here, a language L is called *commutative* if it is closed under commutation, that is, if $w \in L$, then also each permutation of w is in L. Thus, a language L is accepted by a jumping finite automaton if and only if it is the commutative closure of a regular language. In particular, this shows that these languages are rational trace languages, that is, we have the following proper inclusion.

Corollary 4.4. $\mathcal{L}(\mathsf{JFA}) \subsetneq \mathcal{LRAT}$.

By Proposition 3.5, the language $L_\vee$ is not accepted by any nrDFAwtl. However, this language is the commutative closure of the regular language

$$R_\vee = \{ab\}^* \cup \{abb\}^*,$$

hence, it is accepted by a jumping finite automaton. This yields the following incomparability results.

Corollary 4.5. *The language class* $\mathcal{L}(\mathsf{JFA})$ *is incomparable under inclusion to the language classes* $\mathcal{L}(\mathsf{DFAwtl})$ *and* $\mathcal{L}(\mathsf{nrDFAwtl})$.

However, it remains open whether $\mathcal{L}(\mathsf{JFA})$ is contained in the class GCSL of growing context-sensitive languages, or whether there is a language that is accepted by a jumping finite automaton but that is not growing context-sensitive.

Next, we turn to a restricted class of jumping finite automata, the so-called *right one-way jumping finite automaton*.

Definition 4.6. [2, 5] A *nondeterministic right one-way jumping finite automaton*, or an *NROWJFA*, is given through a 5-tuple $J = (Q, \Sigma, I, F, \delta)$, where Q is a finite set of states, Σ is a finite alphabet, $I \subseteq Q$ is the set of initial states, $F \subseteq Q$ is the set of final states, and $\delta : Q \times \Sigma \to \mathcal{P}(Q)$ is a transition relation. For each state $q \in Q$, $\Sigma_q = \{\, a \in \Sigma \mid \delta(q,a) \neq \emptyset \,\}$ is the set of letters that J can read in state q.

A configuration of the NROWJFA J is a word qw from the set $Q \cdot \Sigma^*$. The *computation relation* $\circlearrowright_J^*$ that J induces on its set of configurations is the reflexive and transitive closure of the *right one-way jumping relation* $\circlearrowright_J$ that is defined as follows, where $q, q' \in Q$, $x, y \in \Sigma^*$, and $a \in \Sigma$:

$$qxay \circlearrowright_J q'yx \text{ if } x \in (\Sigma \setminus \Sigma_q)^* \text{ and } q' \in \delta(q,a).$$

Thus, being in state q, J reads and deletes the first letter to the right of the actual head position that it can actually read in that state, while the prefix that consists of letters for which J has no transitions in the current state is cyclically shifted to the end of the current tape contents. Then

$$L(J) = \{\, w \in \Sigma^* \mid \exists\, q_0 \in I\, \exists\, q_f \in F : q_0 w \circlearrowright_J^* q_f \,\}$$

is the language accepted by the NROWJFA J.

The NROWJFA J is *deterministic*, that is, a *right one-way jumping finite automaton* or a *ROWJFA*, if $|I| = 1$ and $|\delta(q,a)| \leq 1$ for all $q \in Q$ and $a \in \Sigma$.

For a ROWJFA, we again use the notation $J = (Q, \Sigma, q_0, F, \delta)$, where q_0 is the only initial state of J and $\delta : Q \times \Sigma \to Q$ is the transition function, that is, we write $\delta(q,a) = q'$ instead of $\delta(q,a) = \{q'\}$.

In [2], Beier and Holzer also consider nondeterministic right one-way jumping finite automata with λ-transitions, that is, these automata can perform a change of state without reading an input letter. Actually, they study three different types of these automata based on the conditions that enable the execution of λ-transitions. However, as finite automata with translucent letters that have λ-transitions have not been defined and investigated yet, we do not consider these types of nondeterministic right one-way jumping automata here.

Now, we illustrate the workings of a (N)ROWJFA with some simple examples taken from [2] and [5].

Example 4.7. Let $J = (\{q_0, q_1\}, \{a, b\}, q_0, \{q_0\}, \delta)$ be a ROWJFA, where $\delta(q_0, a) = q_1$ and $\delta(q_1, b) = q_0$. Then J can execute the following example computation:

$$q_0bbabaa \circlearrowright_J q_1baabb \circlearrowright_J q_0aabb \circlearrowright_j q_1abb \circlearrowright_j q_0ba \circlearrowright_J q_1b \circlearrowright_J q_0,$$

and as q_0 is a final state, we see that the above computation is accepting. In fact, it is easily seen that $L(J) = \{ w \in \{a, b\}^* \mid |w|_a = |w|_b \}$. Actually, it can be shown that the non-context-free language L_{eq3} is accepted by a ROWJFA, too.

Example 4.8. Let $J = (\{q_0, q_1\}, \{a, b\}, q_0, \{q_0, q_1\}, \delta)$ be a ROWJFA, where $\delta(q_0, a) = q_0$, $\delta(q_0, b) = q_1$, and $\delta(q_1, b) = q_1$. Starting in its initial state q_0, J may read some letters a, but once it reads a b, it can only keep on reading b's. It follows that $L(J) = \{a\}^* \cdot \{b\}^*$, a language that is not accepted by any JFA.

Example 4.9. Let $J = (\{p_0, q_0, q_1\}, \{a, b\}, \{p_0, q_0\}, \{p_0, q_0\}, \delta)$ be the NROWJFA that is specified by the following transition relation:

$$\delta(p_0, a) = \{p_0\},\ \delta(q_0, a) = \{q_1\},\ \delta(q_1, b) = \{q_0\}.$$

Starting from state p_0, J accepts on input $w \in \{a, b\}^*$ iff $|w|_b = 0$, while starting from state q_0, J accepts on input $w \in \{a, b\}^*$ iff $|w|_a = |w|_b$. Hence, $L(J) = \{a\}^* \cup \{ w \in \{a, b\}^* \mid |w|_a = |w|_b \} = L'_\vee$.

It has been shown that the language $\{ a^n b^n \mid n \geq 0 \}$ is not accepted by any NROWJFA [1]. Furthermore, the permutation-closed language

$$L'_\vee = \{ w \in \{a, b\}^* \mid |w|_b = 0 \text{ or } |w|_b = |w|_a \}$$

is accepted by a JFA according to Theorem 4.3, but it is not accepted by an ROWJFA [1]. Together with Example 4.8, this shows that the language classes $\mathcal{L}(\mathsf{ROWJFA})$ and $\mathcal{L}(\mathsf{JFA})$ are incomparable under inclusion. Furthermore, as $L'_\vee$ is accepted by the NROWJFA from Example 4.9, we have the following proper inclusion result.

Corollary 4.10. $\mathcal{L}(\mathsf{ROWJFA}) \subsetneq \mathcal{L}(\mathsf{NROWJFA})$.

It is shown in [2] that the language class $\mathcal{L}(\mathsf{JFA})$ coincides with the class of permutation-closed languages that are accepted by NROWJFAs. As already $\mathcal{L}(\mathsf{ROWJFA})$ contains all regular languages, this implies that $\mathcal{L}(\mathsf{JFA})$ is a proper subclass of $\mathcal{L}(\mathsf{NROWJFA})$. Also, it is shown in [2] that all languages accepted by NROWJFAs are semi-linear. Furthermore, it has been observed that the language class $\mathcal{L}(\mathsf{ROWJFA})$ is not closed under union, union with regular sets, intersection, intersection with regular sets, complementation, reversal, product, Kleene star or Kleene plus, and (λ-free) morphisms [1, 5]. In addition, we have the following inclusion results.

Theorem 4.11. $\mathcal{L}(\mathsf{ROWJFA}) \subsetneq \mathcal{L}(\mathsf{nrDFAwtl})$ *and* $\mathcal{L}(\mathsf{NROWJFA}) \subsetneq \mathcal{L}(\mathsf{nrNFAwtl})$.

Proof. As the language $\{ a^n b^n \mid n \geq 0 \}$ is accepted by an nrDFAwtl, while it is not even accepted by any NROWJFA, it follows immediately that the inclusions above are proper inclusions. It remains to verify that the above inclusions actually hold, that is, to show that, for each (N)ROWJFA J, there exists an nrD(N)FAwtl A such that $L(A) = L(J)$.

So let $J = (Q, \Sigma, I, F, \delta)$ be an NROWJFA, and let $\lhd \notin \Sigma$ be a new letter. We now construct a corresponding nrNFAwtl $A = (Q_A, \Sigma, \lhd, \tau, I_A, \delta_A)$ as follows.

Let $Q_A = \{ q, q', \bar{q} \mid q \in Q \}$, $I_A = \{ q' \mid q \in I \}$, and let the translucency mapping τ and the transition relation δ_A be defined through

$$\begin{array}{lcl}
\tau(q) & = & \Sigma \setminus \Sigma_q \text{ for all } q \in Q, \\
\tau(q') & = & \Sigma \setminus \Sigma_q \text{ for all } q \in Q, \\
\tau(\bar{q}) & = & \emptyset \text{ for all } q \in Q, \\
\delta_A(q,a) & = & \delta(q,a) \text{ for all } q \in Q \text{ and all } a \in \Sigma_q, \\
\delta_A(q,\lhd) & = & \{q'\} \text{ for all } q \in Q, \\
\delta_A(q',a) & = & \delta(q,a) \text{ for all } q \in Q \text{ and all } a \in \Sigma_q, \\
\delta_A(q',\lhd) & = & \{\bar{q}\} \text{ for all } q \in Q, \\
\delta_A(\bar{q},\lhd) & = & \begin{cases} \mathsf{Accept}, & \text{if } q \in F, \\ \emptyset, & \text{if } q \notin F. \end{cases}
\end{array}$$

For each state q of the NROWJFA J, the nrNFAwtl A has three corresponding states: q, q', and $\bar{q}$. Here, the states of the form q are used to simulate the corresponding steps of J, while those of the form q' are used to check whether the current tape inscription contains only letters that J cannot process in state q. In that case, state $\bar{q}$ is entered and A accepts if q is a final state of J and if the input has been read (and deleted) completely. Note that the nrNFAwtl A is deterministic, if J is deterministic.

In order to prove that $L(A) = L(J)$, we establish the following two claims.

Claim 1. For all words $w, z, t \in \Sigma^*$ such that $w = zt$ and all states $q \in Q$ and $q_f \in F$, if $qw \circlearrowright_J^* q_f$, then

1. $q'w \cdot \lhd \vdash_A^* \bar{q}_f \cdot \lhd \vdash_A \mathsf{Accept}$ and
2. $tqz \cdot \lhd \vdash_A^* \bar{q}_f \cdot \lhd \vdash_A \mathsf{Accept}$.

Proof. We proceed by induction on the length of the computation of J on input w. So let

$$q_0 w = q_0 w_0 \circlearrowright_J q_1 w_1 \circlearrowright_J \cdots \circlearrowright_J q_{m-1} w_{m-1} \circlearrowright_j q_m w_m = q_f \in F$$

be a computation of the NROWJFA J on an input $w \in \Sigma^m$ starting in a state q_0.

If $m = 0$, then $q_0 = q_f$ and $w = \lambda$. In this situation, we have

1. $q_0' w \cdot \lhd = q_0' \cdot \lhd \vdash_A \bar{q}_0 \cdot \lhd \vdash_A \mathsf{Accept}$ and
2. $z = t = \lambda$ and $tq_0 z \cdot \lhd = q_0 \cdot \lhd \vdash_A q_0' \cdot \lhd = q_0' w \cdot \lhd \vdash_A \bar{q}_0 \cdot \lhd \vdash_A \mathsf{Accept}$,

as $q_0 \in F$.

Now assume that $m \geq 1$, and let us suppose that the claim holds for all computations of J of length less than m. We shall show the claim also holds for all computations of J of length m. Let $q_0 w \circlearrowright_J q_1 vu \circlearrowright_J^{m-1} q_f \in F$, where $w = uav$, $u \in (\Sigma \setminus \Sigma_{q_0})^*$, and $q_1 \in \delta(q_0, a)$. Then $u \in (\tau(q_0'))^*$ and we have

1. $q_0' w \cdot \lhd = q_0' uav \cdot \lhd \vdash_A uq_1 v \cdot \lhd$, where the word vu is of length $m-1$. By applying the induction hypothesis, it follows from $q_1 vu \circlearrowright_J^{m-1} q_f \in F$ that $uq_1 v \cdot \lhd \vdash_A^* \bar{q}_f \cdot \lhd \vdash_A \mathsf{Accept}$.
2. Let $w = zt$ for some words $z, t \in \Sigma^*$. Then either $z = uav_1$ and $v = v_1 t$ or $u = zt_1$ and $t = t_1 av$.
   - If $z = uav_1$ and $v = v_1 t$, then we have

$$tq_0 z \cdot \lhd = tq_0 uav_1 \cdot \lhd \vdash_A tuq_1 v_1 \cdot \lhd.$$

   By the induction hypothesis, it follows from $q_1 vu \circlearrowright_J^{m-1} q_f \in F$ that $tuq_1 v_1 \cdot \lhd \vdash_A^* \bar{q}_f \cdot \lhd \vdash_A \mathsf{Accept}$, as $vu = v_1 tu$.

- If $u = zt_1$ and $t = t_1av$, then we have

$$tq_0z \cdot \lhd = t_1avq_0z \cdot \lhd \vdash_A q_0't_1avz \cdot \lhd \vdash_A t_1q_1vz \cdot \lhd,$$

as z is a prefix of $u \in (\Sigma \setminus \Sigma_{q_0})^* = (\tau(q_0))^*$. By the induction hypothesis, it follows from $q_1vu \circlearrowleft_J^{m-1} q_f \in F$ that $t_1q_1vz \cdot \lhd \vdash_A^* \bar{q}_f \cdot \lhd \vdash_A \mathsf{Accept}$, as $vu = vzt_1$. □

If $w \in L(J)$, then $q_0w \circlearrowleft_J q_f$ for some states $q_0 \in I$ and $q_f \in F$. Now, Claim 1 implies that $q_0'w \cdot \lhd \vdash_A^* \bar{q}_f \cdot \lhd \vdash_A \mathsf{Accept}$. As $q_0' \in I_A$, we have $w \in L(A)$, which proves that $L(J) \subseteq L(A)$.

Claim 2. For all words $w, z, t \in \Sigma^*$ such that $w = zt$ and all states $q_0 \in Q$ and $q_f \in F$, if (1) $q_0'w \cdot \lhd \vdash_A^* \bar{q}_f \cdot \lhd$ or (2) $tq_0z \cdot \lhd \vdash_A^* \bar{q}_f \cdot \lhd$, then $q_0w \circlearrowleft_J^* q_f$.

Proof. We proceed by induction on the length of the input w. So let $w = zt$ and assume that $q_0'w \cdot \lhd \vdash_A^* \bar{q}_f \cdot \lhd$ or $tq_0z \cdot \lhd \vdash_A^* \bar{q}_f \cdot \lhd$ for some states $q_0 \in I$ and $q_f \in F$.

If $|w| = 0$, then $z = \lambda = t$ and $tq_0z \cdot \lhd = q_0 \cdot \lhd \vdash_A q_0' \cdot \lhd \vdash_A \bar{q}_0 \cdot \lhd$, which yields $q_0 = q_f$. Therefore, $q_0w = q_0 \circlearrowleft_J^* q_f$. The case that $q_0'w \cdot \lhd \vdash_A^* \bar{q}_f \cdot \lhd$ is just a part of the above computation, that is, the statement of the claim holds for $|w| = 0$.

Finally, let $|w| = m \geq 1$, and let us suppose that the claim is true for all words of length less than m. Then $w = uav$ for some $u \in (\tau(q_0))^* = (\Sigma \setminus \Sigma_{q_0})^*$, $v \in \Sigma^*$, and a letter $a \in \Sigma$ such that $q_1 \in \delta(q_0, a)$. There are two cases:

1. If $q_0'w \cdot \lhd = q_0'uav \cdot \lhd \vdash_A uq_1v \cdot \lhd \vdash_A^* \bar{q}_f \cdot \lhd$, then $q_0w = q_0uav \circlearrowleft_J q_1vu$. As $|vu| = m - 1$, the induction hypothesis implies that $q_1vu \circlearrowleft_J^* q_f$. Thus, $q_0w \circlearrowleft_J^* q_f$ follows.
2. If $w = zt$ and $tq_0z \cdot \lhd \vdash_A^* \bar{q}_f \cdot \lhd$, we have again two cases: either $z = uav_1$ and $v = v_1t$ or $u = zt_1$ and $t = t_1av$.
   - If $z = uav_1$ and $v = v_1t$, then

$$tq_0z \cdot \lhd = tq_0uav_1 \cdot \lhd \vdash_A tuq_1v_1 \cdot \lhd \vdash_A^* \bar{q}_f \cdot \lhd$$

   and $q_0w = q_0zt = q_0uav_1t \circlearrowleft_J q_1v_1tu$. By applying the induction hypothesis, it follows from $tuq_1v_1 \cdot \lhd \vdash_A^* \bar{q}_f \cdot \lhd$ that $q_1vu \circlearrowleft_J^{m-1} q_f$, as $vu = v_1tu$.
   - If $u = zt_1$ and $t = t_1av$, then

$$tq_0z \cdot \lhd = t_1avq_0z \cdot \lhd \vdash_A q_0't_1avz \cdot \lhd \vdash_A t_1q_1vz \cdot \lhd \vdash_A^* \bar{q}_f \cdot \lhd,$$

   as z is a prefix of $u \in (\Sigma \setminus \Sigma_{q_0})^*$, and $q_0zt = q_0zt_1av \circlearrowleft_J q_1vzt_1$. By applying the induction hypothesis, it follows from $t_1q_1vz \cdot \lhd \vdash_A^* \bar{q}_f \cdot \lhd$ that $q_1vu \circlearrowleft_J^{m-1} q_f \in F$, as $vu = vzt_1$.

   In both cases, we obtain $q_0w \circlearrowleft_J^* q_f$. □

According to the transition relation δ_A, the automaton A can reach an accepting configuration only from a state $\bar{q}_f$ for some $q_f \in F$, and such a state can be reached only after all letters of the input have been deleted. If $w \in L(A)$, then $q_0'w \cdot \lhd \vdash_A^* \bar{q}_f \cdot \lhd \vdash_A \mathsf{Accept}$ for some states $q_0' \in I_A$ and $q_f \in F$. Hence, according to Claim 2, it follows that $q_0w \circlearrowleft_J^* q_f$ and q_f is a final state of J. This proves that $w \in L(J)$ and $L(A) \subseteq L(J)$.

Hence, we see that the inclusions $\mathcal{L}(\mathsf{NROWJFA}) \subseteq \mathcal{L}(\mathsf{nrNFAwtl})$ and $\mathcal{L}(\mathsf{ROWJFA}) \subseteq \mathcal{L}(\mathsf{nrDFAwtl})$ hold. □

When looking at the definitions, we see that the ROWJFA differs from the nrDFAwtl in two aspects. The first of these is the partitioning of the alphabet. For each state q of an ROWJFA J, the alphabet of J is split into two disjoint subsets: the set Σ_q of letters that J can read in state q, and the remaining letters. For a state p of an nrDFAwtl A, the alphabet of A is split into three disjoint subsets: the set of letters that A can read in state p, the set $\tau(p)$ of letters that are translucent for state p, and the set of letters that are neither translucent for state p nor can be read in state p. However, the third type of letters for state p can be avoided by defining, for

each letter b of this type, $\delta_A(p,b) = q_{\mathsf{fail}}$, where q_{fail} is a new state in which A cannot read any letter at all. The second aspect is the fact that an nrDFAwtl has an end-of-tape marker and that, whenever its head reaches this marker, then A can execute an additional change of state. It is actually this feature that allows the nrDFAwtl to accept the language $\{ a^n b^n \mid n \geq 0 \}$, that is, the increase in expressive capacity from the ROWJFA to the nrDFAwtl is due to this second feature.

It remains to compare the NROWJFA and the ROWJFA to the DFAwtl and to the NFAwtl. According to [2], the language

$$L = \{ wa \mid w \in \{a,b\}^*, |w|_a = |w|_b \}$$

is accepted by a DFAwtl, but it is not accepted by any NROWJFA, which means that $\mathcal{L}(\mathsf{DFAwtl})$ is not contained in $\mathcal{L}(\mathsf{NROWJFA})$. On the other hand, there it is shown that the language

$$L = \{ ucvcw \mid u,v,w \in \{a,b\}^*, |uw|_a = |uw|_b \}$$

is accepted by an ROWJFA, but not by any NFAwtl. Therefore, we have the following incomparability result.

Corollary 4.12. [2] *The language classes $\mathcal{L}(\mathsf{DFAwtl})$ and $\mathcal{L}(\mathsf{NFAwtl})$ are incomparable under inclusion to the language classes $\mathcal{L}(\mathsf{ROWJFA})$ and $\mathcal{L}(\mathsf{NROWJFA})$.*

Next, we compare the right one-way jumping automaton to the class GCSL of growing context-sensitive languages.

Theorem 4.13. *There exists a ROWJFA A such that the language $L(A)$ is not growing context-sensitive.*

Proof. Let $A = (Q, \Sigma, q_0, F, \delta)$ be the ROWJFA that is defined as follows:

- $Q = \{q_0, q_a, q_b, q_-\}$ and $F = \{q_0\}$,
- $\Sigma = \{a, b, a', b'\}$,
- $\delta(q_0,a) = q_a$, $\delta(q_0,b) = q_b$,
  $\delta(q_a,a') = q_0$, $\delta(q_a,b') = q_-$,
  $\delta(q_b,a') = q_-$, $\delta(q_b,b') = q_0$.

We claim that $L(A) \cap (\{a',b'\}^* \cdot \{a,b\}^*) = \{ \varphi(w)w \mid w \in \{a,b\}^* \}$, where $\varphi : \{a,b\}^* \to \{a',b'\}^*$ is the alphabetic morphism that is specified through the mapping $a \mapsto a'$ and $b \mapsto b'$.

Claim 1. For each word $w \in \{a,b\}^*$, $\varphi(w)w \in L(A)$.

Proof. By induction on the length of the word $w \in \{a,b\}^*$, we show that $q_0\varphi(w)w \circlearrowright_A^* q_0$, hence, A accepts $\varphi(w)w$.

If $|w| = 0$, then $q_0\varphi(w)w = q_0 \in F$ and we see that $w = \lambda \in L(A)$.

If $w = cx$ for some letter $c \in \{a,b\}$ and a word $x \in \{a,b\}^*$, then

$$q_0\varphi(w)w = q_0c'\varphi(x)cx \circlearrowright_A q_cxc'\varphi(x) \circlearrowright_A q_0\varphi(x)x,$$

and induction shows that $q_0\varphi(x)x \circlearrowright_A^* q_0 \in F$. Thus, $\varphi(w)w \in L(A)$ follows. □

Hence, we see that $\{ \varphi(w)w \mid w \in \{a,b\}^* \} \subseteq L(A)$.

Claim 2. If $x \in \{a',b'\}^*$ and $y \in \{a,b\}^*$ such that $xy \in L(A)$, then $x = \varphi(y)$.

Proof. In each accepting computation of the automaton A, the state q_0 alternates with the states q_a and q_b. That is, after an even number of steps, the automaton is in state q_0, and after an odd number of steps, the automaton is either in state q_a or in state q_b. In state q_0, only an a or a b can be read. From state q_a (q_b), a letter a' (b') will take A back to state q_0, while a letter b' (a') takes A to the trap state q_-. As q_0 is the only

accepting state of A, in every two consecutive steps of an accepting computation of A, one letter is read from $\{a, b\}$ and one letter is read from $\{a', b'\}$. The computation starts and ends in state q_0, hence, each accepted word must contain the same number of letters from $\{a, b\}$ as it contains letters from $\{a', b'\}$. As $xy \in L(A)$, this means that x and y are of the same length. We claim that $x = \varphi(y)$.

In order to derive a contradiction, we assume that $xy \in L(A)$, $|x| = |y|$, and $x \neq \varphi(y)$. Then $x = \varphi(u)c'v'$ and $y = udz$, where $u, z \in \{a, b\}^*$, $c, d \in \{a, b\}$, $c \neq d$, and $v' \in \{a', b'\}^*$. If $u = \lambda$, then $q_0xy = q_0c'v'dz \circlearrowright_A q_dzc'v' \circlearrowright_A q_-v'z$, and the automaton rejects the word xy.

If $u \neq \lambda$, we have

$$q_0xy = q_0\varphi(u)c'v'udz \circlearrowright_A^* q_0c'v'dz \circlearrowright_A q_dzc'v' \circlearrowright_A q_-v'z,$$

and the computation fails.

In both cases, the word xy is rejected – a contradiction to the assumption that $xy \in L(A)$. Thus, we can conclude that, if $xy \in L(A)$, then $x = \varphi(y)$. □

Together, the two claims above prove that

$$L(A) \cap (\{a', b'\}^* \cdot \{a, b\}^*) = \{\, \varphi(w)w \mid w \in \{a, b\}^* \,\}.$$

It remains to argue that the language $L(A)$ is not growing context-sensitive. Assume to the contrary that the language $L(A)$ is growing context-sensitive. As the class GCSL of growing context-sensitive languages is closed under intersection with regular languages, it then follows that the language

$$L(A) \cap (\{a', b'\}^* \cdot \{a, b\}^*) = \{\, \varphi(w)w \mid w \in \{a, b\}^* \,\}$$

is growing context-sensitive. However, the class GCSL is closed under alphabetic morphisms, this contradicts the fact that the copy language $L_{\text{copy}} = \{\, ww \mid w \in \{a, b\}^* \,\}$ is not growing context-sensitive [4]. Thus, we see that the language $L(A)$ is indeed not growing context-sensitive. □

Finally, following [5], we also relate the various types of finite automata with translucent letters to the right-revolving finite automaton of [3].

Definition 4.14. [3] A *right-revolving NFA*, or an *rr-NFA*, is given through a 6-tuple $A = (Q, \Sigma, q_0, F, \Delta, \delta)$, where Q is a finite set of states, Σ is a finite alphabet, $q_0 \in Q$ is an initial state, $F \subseteq Q$ is the set of final states, and $\Delta : Q \times \Sigma \to \mathcal{P}(Q)$ and $\delta : Q \times \Sigma \to \mathcal{P}(Q)$ are two transition relations.

A configuration of the rr-NFA A is a word qw, where $q \in Q$ and $w \in \Sigma^*$. The *computation relation* $\vdash_A$ that A induces on its set of configurations is the reflexive and transitive closure of the *one-step computation relation* $\vdash_A$ that is defined as follows, where $a \in \Sigma$ and $q, q' \in Q$:

1. If $q' \in \delta(q, a)$, then $qaw \vdash_A q'w$. The transitions of this form are called *ordinary transitions*.
2. If $q' \in \Delta(q, a)$, then $qaw \vdash_A q'wa$. The transitions of this form are called *right-revolving transitions*.

Thus, an ordinary transition consumes the first letter of the current tape contents, while a right-revolving transition just shifts the first letter to the end of the tape. Observe that it is possible that $\delta(q, a) \neq \emptyset \neq \Delta(q, a)$. In this case, A chooses nondeterministically whether to execute an ordinary or a right-revolving transition. Then

$$L(A) = \{\, w \in \Sigma^* \mid \exists q_f \in F : q_0w \vdash_A^* q_f \,\}$$

is the language accepted by the rr-NFA A.

The rr-NFA A is *deterministic*, or an *rr-DFA*, if, for all $q \in Q$ and all $a \in \Sigma$, $|\Delta(q, a)| + |\delta(q, a)| \leq 1$.

In [3], also λ-transitions are defined for rr-NFAs and rr-DFAs, but it is proved in Theorem 5 of that paper that λ-transitions do not increase the expressive capacity of right-revolving finite automata. Accordingly, we do

not introduce these λ-transitions in the first place. Concerning the expressive capacity of right-revolving finite automata, the following results have been obtained.

Proposition 4.15. [3, 5]

(a) $\mathcal{L}(\mathsf{ROWJFA}) \subsetneq \mathcal{L}(\mathsf{rr\text{-}DFA})$.
(b) *A unary language L is accepted by an rr-NFA iff L is regular.*
(c) *The non-context-free language*

$$L_{eq3} = \{ w \in \{a,b,c\}^* \mid |w|_a = |w|_b = |w|_c \}$$

is accepted by an rr-DFA.
(d) *The deterministic linear context-free language $L = \{ a^n b^n \mid n \geq 0 \}$ is not accepted by any rr-NFA.*
(e) $L = \{a\}^+ \cup \{ w \in \{a,b\}^* \mid |w|_a = |w|_b \} \in \mathcal{L}(\mathsf{rr\text{-}NFA}) \setminus \mathcal{L}(\mathsf{rr\text{-}DFA})$.

The proof of the inclusion in (a) given in [5] easily extends to the nondeterministic case, that is, we also have $\mathcal{L}(\mathsf{NROWJFA}) \subseteq \mathcal{L}(\mathsf{rr\text{-}NFA})$. Below, we present a separating example language (see Cor. 4.19).

It follows from Theorem 4.13 that none of the language classes $\mathcal{L}(\mathsf{NROWJFA})$, $\mathcal{L}(\mathsf{rr\text{-}DFA})$, and $\mathcal{L}(\mathsf{rr\text{-}NFA})$ is contained in the class GCSL, either. On the other hand, as GCSL contains all context-free languages, the fact that $\{ a^n b^n \mid n \geq 0 \} \notin \mathcal{L}(\mathsf{rr\text{-}NFA})$ implies that the class GCSL is not contained in any of these language classes. Thus, we have the following incomparability result.

Corollary 4.16. *The language classes $\mathcal{L}(\mathsf{ROWJFA})$, $\mathcal{L}(\mathsf{NROWJFA})$, $\mathcal{L}(\mathsf{rr\text{-}DFA})$, and $\mathcal{L}(\mathsf{rr\text{-}NFA})$ are incomparable to the class GCSL with respect to inclusion.*

It remains open whether the class $\mathcal{L}(\mathsf{JFA})$ is contained in GCSL.

Next, we introduce the language L_{cc}, which can be seen as a simple variant of the language $\{ \varphi(w)w \mid w \in \{a,b\}^* \}$ considered above. The language L_{cc} is defined as follows:

$$L_{cc} = \{ wcwc \mid w \in \{a\}^* \} = \{ a^m c a^m c \mid m \geq 0 \}.$$

First, we prove that a deterministic right-revolving finite automaton accepts a certain superset of this language.

Proposition 4.17. *There exists an rr-DFA A such that*

$$L_{cc} = L(A) \cap (\{a\}^* \cdot \{c\} \cdot \{a\}^* \cdot \{c\}).$$

Proof. Let $A = (Q, \Sigma, q_0, F, \Delta, \delta)$ be the rr-DFA that is defined as follows:

- $Q = \{q_0, q_1, q_a, q_c, q'_a, q_+, q_-\}$ and $F = \{q_+\}$,
- $\Sigma = \{a, c\}$,
- $\Delta(q_a, a) = q_a$, $\Delta(q_a, c) = q'_a$,
  $\Delta(q_1, a) = q_1$, $\Delta(q_1, c) = q_0$,
- $\delta(q_0, a) = q_a$, $\delta(q_0, c) = q_c$,
  $\delta(q'_a, a) = q_1$, $\delta(q'_a, c) = q_-$,
  $\delta(q_c, a) = q_-$, $\delta(q_c, c) = q_+$.

This rr-DFA accepts a word of the form $awcawc$, where $w \in \{a\}^*$, as follows:

$$\begin{array}{lllllll} q_0awcawc & \vdash_A & q_awcawc & \vdash_A^{|w|} & q_acawcw & \vdash_A & q'_awcwc \\ & \vdash_A & q_1wcwc & \vdash_A^{|w|} & q_1cwcw & \vdash_A & q_0wcwc \\ & \vdash_A^* & q_0cc & \vdash_A & q_cc & \vdash_A & q_+. \end{array}$$

On the other hand, it is easily seen that any word of the form $a^m c a^n c$, where $m \neq n$, is not accepted by A. Thus, it follows that

$$L(A) \cap (\{a\}^* \cdot \{c\} \cdot \{a\}^* \cdot \{c\}) = L_{cc}$$

as claimed. □

We now prove that there does not exist any nondeterministic right one-way jumping automaton for the language L_{cc}. Actually, we have the following stronger negative result.

Proposition 4.18. *There does not exist any nrNFAwtl C such that*

$$L(C) \cap (\{a\}^* \cdot \{c\} \cdot \{a\}^* \cdot \{c\}) = L_{cc}.$$

Proof. Assume that $C = (Q, \Sigma, \lhd, \tau, I, \delta)$ is an nrNFAwtl such that $L(C) \cap (\{a\}^* \cdot \{c\} \cdot \{a\}^* \cdot \{c\}) = L_{cc}$. We derive a contradiction by analyzing accepting computations of C on inputs of the form $w_m = a^m c a^m c \in L_{cc}$, where $m \geq 1$.

Let $m \geq 1$ be a (sufficiently) large integer. As $w_m = a^m c a^m c \in L_{cc}$, the nrNFAwtl C has an accepting computation for the input w_m, which looks as follows, where $q_0 \in I$:

$$q_0 w \cdot \lhd = q_0 a^m c a^m c \cdot \lhd \vdash_C^* q_+ \cdot \lhd \vdash_C \mathsf{Accept},$$

where $q_+ \in Q$ and $\delta(q_+, \lhd) = \mathsf{Accept}$. Recall from Lemma 2.9 that we may assume without loss of generality that C only accepts once it has read (and deleted) the given input completely.

We can assume that the above computation begins with a sequence of $r \geq 0$ steps in which occurrences of the letter a are read:

$$\begin{array}{lllllll} q_0 a^m c a^m c \cdot \lhd & \vdash_C & q_1 a^{m-1} c a^m c \cdot \lhd & \vdash_C & q_2 a^{m-2} c a^m c \cdot \lhd & \vdash_C & \cdots \\ & \vdash_C & q_r a^{m-r} c a^m c \cdot \lhd & \vdash_C^* & q_+ \cdot \lhd & \vdash_C & \mathsf{Accept}, \end{array}$$

where $q_1, q_2, \ldots, q_r \in Q$. If $r \geq |Q|$, then there are indices i, j, $0 \leq i < j \leq r$, such that $q_i = q_j$. In this case, the above computation looks as follows:

$$\begin{array}{lllll} q_0 a^m c a^m c \cdot \lhd & \vdash_C^i & q_i a^{m-i} c a^m c \cdot \lhd & \vdash_C^{j-i} & q_j a^{m-j} c a^m c \cdot \lhd \\ & = & q_i a^{m-j} c a^m c \cdot \lhd & \vdash_C^* & q_+ \cdot \lhd \\ & \vdash_C & \mathsf{Accept}. & & \end{array}$$

However, C can then also execute the following computation:

$$q_0 a^{m-j+i} c a^m c \cdot \lhd \quad \vdash_C^i \quad q_i a^{m-j} c a^m c \cdot \lhd \quad \vdash_C^* \quad q_+ \cdot \lhd \quad \vdash_C \quad \mathsf{Accept},$$

that is, C accepts the word $a^{m-j+i} c a^m c \in (\{a\}^* \cdot \{c\} \cdot \{a\}^* \cdot \{c\}) \smallsetminus L_{cc}$, a contradiction. Thus, $r < |Q|$ follows, which in turn implies that $a \in \tau(q_r)$.

If $c \in \tau(q_r)$, then C has to apply a transition $q' \in \delta(q_r, \lhd)$ in the configuration $q_r a^{m-r} c a^m c \cdot \lhd$, which yields the configuration $q' a^{m-r} c a^m c \cdot \lhd$. Now, further occurrences of the letter a could be read, but the pumping argument above shows that the number of these steps is smaller than $|Q| - r$. Hence, we finally obtain a configuration of the form $q a^s c a^m c \cdot \lhd$, where $s \leq m$ and $q \in Q$ such that $a \in \tau(q)$, but $c \notin \tau(q)$.

Then, the above computation continues as follows, where $q' \in \delta(q, c)$:

$$q a^s c a^m c \cdot \lhd \vdash_C a^s q' a^m c \cdot \lhd \vdash_C^* q_+ \cdot \lhd \vdash_C \mathsf{Accept}.$$

Next, t occurrences of the letter a may be deleted from the factor a^m, but by using the same pumping argument as above, it follows that the number t is bounded from above by $|Q|$, that is,

$$a^s q' a^m c \cdot \lhd \vdash_C^t a^s q'' a^{m-t} c \cdot \lhd \vdash_C^* q_+ \cdot \lhd,$$

and $a \in \tau(q'')$. Hence, the computation continues with

$$a^s q'' a^{m-t} c \cdot \lhd \vdash_C \hat{q} a^s a^{m-t} c^\varepsilon \cdot \lhd \vdash_C^* q_+ \cdot \lhd,$$

where either $c \in \tau(q'')$, $\varepsilon = 1$, and $\hat{q} \in \delta(q'', \lhd)$, or $c \notin \tau(q'')$, $\varepsilon = 0$, $\tilde{q} \in \delta(q'', c)$, and $\hat{q} \in \delta(\tilde{q}, \lhd)$ for some state $\tilde{q} \in Q$. However, in this case, C can also execute the following computation:

$$\begin{array}{lllll} q_0 a^{m+1} c a^{m-1} c \cdot \lhd & \vdash_C^* & q a^{s+1} c a^{m-1} c \cdot \lhd & \vdash_C & a^{s+1} q' a^{m-1} c \cdot \lhd \\ & \vdash_C^t & a^{s+1} q'' a^{m-1-t} c \cdot \lhd & \vdash_C & \hat{q} a^{s+1} a^{m-1-t} c^\varepsilon \cdot \lhd \\ & = & \hat{q} a^s a^{m-t} c^\varepsilon \cdot \lhd & \vdash_C^* & q_+ \cdot \lhd \\ & \vdash_C & \mathsf{Accept}, & & \end{array}$$

that is, C accepts the word $a^{m+1} c a^{m-1} c \in (\{a\}^* \cdot \{c\} \cdot \{a\}^* \cdot \{c\}) \setminus L_{cc}$, a contradiction.

It follows that indeed $L(C) \cap (\{a\}^* \cdot \{c\} \cdot \{a\}^* \cdot \{c\}) \neq L_{cc}$. □

Observe that the above proof shows, in particular, that the deterministic linear language $\{ a^m c a^m c \mid m \geq 0 \}$ is not accepted by any nrNFAwtl. Together, Proposition 4.17 and Proposition 4.18 yield the following separation result, as by Theorem 4.11, $\mathcal{L}(\mathsf{NROWJFA}) \subsetneq \mathcal{L}(\mathsf{nrNFAwtl})$.

Corollary 4.19. *The language class $\mathcal{L}(\mathsf{NROWJFA})$ is a proper subclass of the language class $\mathcal{L}(\mathsf{rr\text{-}NFA})$.*

In addition, these propositions also show that $\mathcal{L}(\mathsf{rr\text{-}DFA})$ and $\mathcal{L}(\mathsf{rr\text{-}NFA})$ are not contained in $\mathcal{L}(\mathsf{nrDFAwtl})$ or $\mathcal{L}(\mathsf{nrNFAwtl})$. Moreover, in analogy to Example 2.7, the language $\{ a^m c b^m c \mid m \geq 0 \}$ can be shown to be accepted by an nrDFAwtl. If $\varphi : \{a, b, c\}^* \to \{a, c\}^*$ is the morphism defined through $a \mapsto a$, $b \mapsto a$, and $c \mapsto c$, then $\varphi(\{ a^m c b^m c \mid m \geq 0 \}) = \{ a^m c a^m c \mid m \geq 0 \}$. Hence, we obtain the following non-closure result.

Corollary 4.20. *The language class $\mathcal{L}(\mathsf{nrNFAwtl})$ is not closed under alphabetic morphisms.*

Finally, we consider the language $L_c = \{ wc \mid w \in \{a, b\}^*, |w|_a = |w|_b \}$.

Lemma 4.21. $L_c \in \mathcal{L}(\mathsf{DFAwtl})$.

Proof. Let $A = (Q, \Sigma, \lhd, \tau, q_0, F, \delta)$ be the DFAwtl that is defined as follows:

- $Q = \{q_0, q_a, q_b, q_c\}$, $\Sigma = \{a, b, c\}$, and $F = \{q_c\}$,
- $\tau(q_0) = \emptyset$, $\tau(q_a) = \{a\}$, $\tau(q_b) = \{b\}$, and $\tau(q_c) = \emptyset$,
- $\delta(q_0, a) = q_a$, $\delta(q_0, b) = q_b$, $\delta(q_0, c) = q_c$,
  $\delta(q_a, b) = q_0$, $\delta(q_b, a) = q_0$.

Starting in its initial state q_0, A reads the first letter x on its tape. If this letter is an a (b), then A enters the state q_a (q_b), in which it looks for the first occurrence of the letter b (a) that is only preceded by occurrences of the letter a (b). If such a letter is found, then it is read (and deleted), and A returns to its initial state – otherwise, it gets stuck. Finally, if an occurrence of the letter c is read in state q_0, then A halts immediately, and it accepts iff the input has been read completely, which means that there is only a single occurrence of the letter c, which is the last letter of the given input. It follows that, indeed, $L(A) = L_c$. □

On the other hand, we have the following negative result.

Proposition 4.22. *The language L_c is not accepted by any rr-NFA.*

Proof. Assume to the contrary that $A = (Q, \Sigma, q_0, F, \Delta, \delta)$ is an rr-NFA such that $L(A) = L_c$. We derive a contradiction by showing that, together with the words of the language L_c, the rr-NFA A also accepts some words that are not elements of L_c.

For $m \geq 1$, let $w_m = a^m b^{2m} a^m c \in L_c$. Then A has an accepting computation for input w_m, which begins as follows:

$$q_0 w_m = q_0 a^m b^{2m} a^m c \vdash_A^m p_1 b^{2m} a^m c a^k,$$

where k is an integer satisfying $0 \leq k \leq m$. During these first m steps, A consumes $m - k$ occurrences of the letter a by applying its transition function δ, and it shifts k occurrences of the letter a to the end of the tape by applying the function Δ. For a large value of m, we now analyze these first m steps in detail by differentiating between several cases.

Case 1: There exists a continuous subsequence of $r \geq |Q|$ steps such that, in each of them, an occurrence of the letter a is read (and deleted). Then, the above initial part of the accepting computation of A on input w_m can be written as follows:

$$\begin{array}{lcll} q_0 w_m & = & q_0 a^m b^{2m} a^m c & \vdash_A^{i_1} \; q_1 a^{m-i_1} b^{2m} a^m c a^{j_1} \\ & \vdash_A & q_2 a^{m-i_1-1} b^{2m} a^m c a^{j_1} & \vdash_A^{r-2} \; q_r a^{m-i_1-(r-1)} b^{2m} a^m c a^{j_1} \\ & \vdash_A & q_{r+1} a^{m-i_1-r} b^{2m} a^m c a^{j_1} & \vdash_A^* \; p_1 b^{2m} a^m c a^k, \end{array}$$

where $0 \leq j_1 \leq i_1$. As $r \geq |Q|$, there exist indices $1 \leq s < t \leq r+1$ such that $q_s = q_t$, that is, we have

$$\begin{array}{lcl} q_s a^{m-i_1-(s-1)} b^{2m} a^m c a^{j_1} & \vdash_A^{t-s} & q_t a^{m-i_1-(t-1)} b^{2m} a^m c a^{j_1} \\ & = & q_s a^{m-i_1-(t-1)} b^{2m} a^m c a^{j_1}. \end{array}$$

Now we take the input word $y_{m+t-s} = a^{m+t-s} b^{2m} a^m c$. Observe that $y_{m+t-s} \notin L_c$, as $t - s > 0$. However, A can execute the following computation on input y_{m+t-s}:

$$\begin{array}{lcl} q_0 y_{m+t-s} & = & q_0 a^{m+t-s} b^{2m} a^m c \\ & \vdash_A^{i_1} & q_1 a^{m+t-s-i_1} b^{2m} a^m c a^{j_1} \\ & \vdash_A^{s-1} & q_s a^{m+t-s-i_1-(s-1)} b^{2m} a^m c a^{j_1} \\ & \vdash_A^{t-s} & q_t a^{m+t-s-i_1-(t-1)} b^{2m} a^m c a^{j_1} \\ & = & q_s a^{m-i_1-(s-1)} b^{2m} a^m c a^{j_1}, \end{array}$$

and the latter configuration occurs in the accepting computation of A on input w_m. Thus, with the word w_m, the rr-NFA A also accepts the word $y_{m+t-s} \notin L_c$.

Case 2: There does not exist a continuous subsequence of $r \geq |Q|$ steps such that, in each of them, an occurrence of the letter a is read (and deleted), but there is a continuous subsequence of $r \geq |Q|$ steps such that, in each of them, an occurrence of the letter a is rotated to the end of the tape. Then, the initial part of the above accepting computation of A on input w_m can be written as follows:

$$\begin{array}{lcll} q_0 w_m & = & q_0 a^m b^{2m} a^m c & \vdash_A^{i_1} \; q_1 a^{m-i_1} b^{2m} a^m c a^{j_1} \\ & \vdash_A & q_2 a^{m-i_1-1} b^{2m} a^m c a^{j_1+1} & \vdash_A^{r-2} \; q_r a^{m-i_1-(r-1)} b^{2m} a^m c a^{j_1+r-1} \\ & \vdash_A & q_{r+1} a^{m-i_1-r} b^{2m} a^m c a^{j_1+r} & \vdash_A^* \; p_1 b^{2m} a^m c a^k, \end{array}$$

where $0 \leq j_1 \leq i_1$. As $r \geq |Q|$, there exist indices $1 \leq s < t \leq r+1$ such that $q_s = q_t$, that is, we have

$$\begin{array}{lcl} q_s a^{m-i_1-(s-1)} b^{2m} a^m c a^{j_1+s-1} & \vdash_A^{t-s} & q_t a^{m-i_1-(t-1)} b^{2m} a^m c a^{j_1+t-1} \\ & = & q_s a^{m-i_1-(t-1)} b^{2m} a^m c a^{j_1+t-1}. \end{array}$$

Now we take the input $z_{m,t-s} = a^{m-(t-s)}b^{2m}a^m ca^{t-s}$. Observe that $z_{m,t-s} \notin L_c$, as $t-s > 0$. However, A can execute the following computation on input $z_{m,t-s}$:

$$\begin{array}{rcl} q_0 z_{m,t-s} & = & q_0 a^{m-(t-s)}b^{2m}a^m ca^{t-s} \\ & \vdash_A^{i_1} & q_1 a^{m-(t-s)-i_1}b^{2m}a^m ca^{t-s+j_1} \\ & \vdash_A^{s-1} & q_s a^{m-(t-s)-i_1-(s-1)}b^{2m}a^m ca^{t-s+j_1+s-1} \\ & = & q_t a^{m-i_1-(t-1)}b^{2m}a^m ca^{j_1+t-1}, \end{array}$$

and the latter configuration occurs in the accepting computation of A on input w_m. Thus, with the word w_m, the rr-NFA A also accepts the word $z_{m,t-s} \notin L_c$.

Case 3: There is neither a continuous subsequence of $r \geq |Q|$ steps such that, in each of them, an occurrence of the letter a is read (and deleted) nor a continuous subsequence of $r \geq |Q|$ steps such that, in each of them, an occurrence of the letter a is rotated to the end of the tape. This means that, if m is sufficiently large, then subsequences of deletions of occurrences of the letter a and subsequences of rotations of occurrences of the letter a alternate more than $|Q|$ times. Thus, there exists a state $q \in Q$ such that the above initial part of the accepting computation of A on input w_m contains a subcomputation of the following form:

$$qa^{m-i_2}b^{2m}a^m ca^{j_2} \vdash_A^r qa^{m-i_2-r}b^{2m}a^m ca^{j_2+s},$$

where $1 \leq s < r$. Now we take the input $z_{m,r,s} = a^{m-r}b^{2m}a^m ca^s$. Obviously, $z_{m,r,s} \notin L_c$, as $s \geq 1$ and $r > 1$. However, A can execute the following computation on input $z_{m,r,s}$:

$$\begin{array}{rcl} q_0 z_{m,r,s} & = & q_0 a^{m-r}b^{2m}a^m ca^s \\ & \vdash_A^* & qa^{m-r-i_2}b^{2m}a^m ca^{s+j_2} \\ & = & qa^{m-i_2-r}b^{2m}a^m ca^{j_2+s}, \end{array}$$

and the latter configuration occurs in the accepting computation of A on input w_m. Thus, with the word w_m, the rr-NFA A also accepts the word $z_{m,r,s} \notin L_c$.

As this covers all possibilities, it follows that $L(A) \neq L_c$, completing the proof of Proposition 4.22. □

Together, Lemma 4.21 and Proposition 4.22 imply that $\mathcal{L}(\mathsf{DFAwtl})$ is not contained in $\mathcal{L}(\mathsf{rr\text{-}NFA})$. Together with Proposition 4.17 and Proposition 4.18, this yields the following incomparability result.

Corollary 4.23. *The language classes* $\mathcal{L}((\mathsf{nr})\mathsf{DFAwtl})$ *and* $\mathcal{L}((\mathsf{nr})\mathsf{NFAwtl})$ *are incomparable under inclusion to the language classes* $\mathcal{L}(\mathsf{rr\text{-}DFA})$ *and* $\mathcal{L}(\mathsf{rr\text{-}NFA})$.

Observe that the language L_c is a rational trace language. In fact,

$$L_c = \bigcup_{w \in R} [w]_D$$

for the regular language $R = \{ab\}^* \cdot \{c\}$ and the dependency relation $D = \{(a,a),(a,c),(b,b),(b,c),(c,a),(c,b),(c,c)\}$. Hence, we see that $\mathcal{L}(\mathsf{rr\text{-}NFA})$ does not contain all rational trace languages. However, as for nrNFAwtl and nrDFAwtl, it seems to be still open whether rr-DFA and rr-NFA only accept languages that are semi-linear.

The diagram in Figure 2 summarizes the inclusion and incomparability results obtained. All arrows in that diagram denote proper inclusions, and classes that are not connected by a sequence of arrows are incomparable under inclusion. The only possible exception is the question of whether $\mathcal{L}(\mathsf{JFA})$ is contained in GCSL.

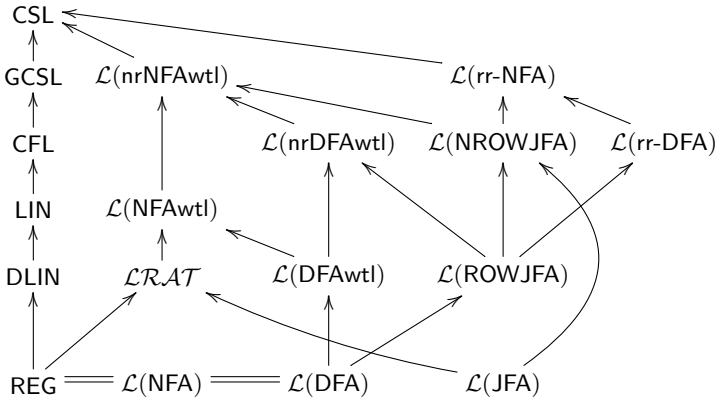


FIGURE 2. The inclusion relations between the various types of jumping automata, right-revolving automata, and automata with translucent letters.

5. The fixed membership problem for the nrDFAwtl

The fixed membership problem is certainly the most fundamental among the decision problems for automata and languages. Let A be an automaton of a certain fixed type, *e.g.*, a DFA or an nrDFAwtl, with an input alphabet Σ. Then the *membership problem for* A can be stated as follows:

INSTANCE : A word $w \in \Sigma^*$.
QUESTION : Is w an element of the language $L(A)$?

The membership problem for each nrDFAwtl is solvable in linear space, as an nrDFAwtl can easily be simulated by a deterministic linear-bounded automaton. Moreover, it is straightforward to see that $\mathcal{L}(\mathsf{nrDFAwtl}) \subseteq \mathsf{DTIME}(n^2)$, since an nrDFAwtl can also be simulated by a two-tape Turing machine in quadratic time, that is, the membership problem for an nrDFAwtl is decidable in quadratic time. In fact, we can do better than that. If A is an nrDFAwtl over a unary alphabet, then, by Proposition 3.3, the language accepted by A is necessarily regular. Hence, we can use a DFA to solve the corresponding membership problem, that is, this problem is decidable in linear time.

In [16], Nagy and Kovács present a construction that shows that the membership problem for a DFAwtl is solvable in time $\mathcal{O}(n \cdot \log n)$ by a random access machine (a RAM) using the logarithmic complexity measure. For a given input word w, they first construct a specific data structure that is then used to simulate the computation of the given DFAwtl on input w. By using a variation of that technique, we derive the following result.

Theorem 5.1. *The membership problem for an nrDFAwtl over a binary alphabet is decidable in time* $\mathcal{O}(n \cdot \log n)$.

Proof. Let $A = (Q, \Sigma, \lhd, \tau, q_0, \delta)$ be an nrDFAwtl such that $|\Sigma| = 2$. By Lemma 2.9, we can assume that A never gets into an infinite computation and that it accepts only after reading and deleting its input completely. From the proof of that lemma, we can moreover conclude that, for a word w of length n, the computation of A on input w consists of only $\mathcal{O}(n)$ many steps. Furthermore, for all states $q \in Q$ and all letters $c \in \Sigma \cup \{\lhd\}$, if $c \notin \tau(q)$ and if $\delta(q, c)$ is undefined, then we extend the transition function δ by taking $\delta(q, c) = \mathsf{Reject}$.

For the following considerations, we assume for simplicity that $\Sigma = \{a, b\}$. Therefore, each input word $w \in \Sigma^+$ can be written as

$$w = a^{i_1} b^{i_2} a^{i_3} \cdots b^{i_{2m}} a^{i_{2m+1}},$$

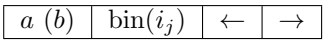


FIGURE 3. The graphical representation of a record of type *block*.

where $m \geq 0$, $i_1 \geq 0$, $i_2, i_3, \ldots, i_{2m} \geq 1$, and $i_{2m+1} \geq 0$. For each $1 \leq j \leq 2m+1$, if j is odd, then the factor a^{i_j} is called an *a-block* of w, and if j is even, then the factor b^{i_j} is called a *b-block* of w. Now, for all $j = 1, 2, \ldots, 2m+1$, we realize a record rec_j of type *block* that records the j-th block of w:

type *block* = **record**
symbol : *char from* $\{\triangleright, a, b, \triangleleft\}$,
length : $\mathbb{N}$,
prev, *next* : $\uparrow$ *block*
end

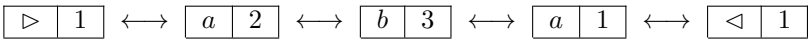
Here the component *symbol* gives the type of the block, where $\triangleright$ ($\triangleleft$) is used for the block that symbolizes the beginning (end) of the input, the component *length* encodes the length of the current block, and *prev* and *next* are pointers to the records of type *block* that correspond to the previous block and the next block. Obviously, a natural number like *length* is represented in binary, and pointers like *prev* and *next* are represented as addresses coded in binary. Thus, such a record of type block can be represented by a diagram as in Figure 3.

The algorithm for solving the membership problem for the nrDFAwtl A proceeds in two major steps:

1. Let $w \in \Sigma^*$ be given as input. If $w = \lambda$, then the algorithm accepts immediately if $\lambda \in L(A)$; otherwise, it rejects. If $w \neq \lambda$, then a representation of $w \cdot \triangleleft$ as a sequence of records of type *block* is constructed.
2. Using the sequence of records of type *block* constructed in step 1, the computation of the nrDFAwtl A on input w is simulated. The algorithm accepts if this computation accepts; otherwise, it rejects.

This algorithm uses two procedures. The first of them, Build_List_of_Blocks, generates the representation of the word $w \cdot \triangleleft$, while the second one, Simulate_nrDFAwtl, uses that representation to determine whether w is accepted by the nrDFAwtl A.

For example, if $w = aabbba$, then the procedure Build_List_of_Blocks generates the following doubly-linked list, where we represent the various pointers by double arrows:



It is easily seen that, for an input of the form $w \cdot \triangleleft$, where $w \in \{a, b\}^n$, the procedure Build_List_of_Blocks executes $\mathcal{O}(n)$ many steps.

The procedure Simulate_nrDFAwtl keeps the current state q, the current block B, and the position r of the head of the nrDFAwtl A within the current block. Whenever A reads and deletes a letter from B, then the length of B is reduced by one. If block B is emptied during that process, then this block is cut out from the sequence, and its predecessor C and its successor D are merged. In this situation, the head of the nrDFAwtl A is placed on the r-th letter of this new block, where $r = C.length + 1$. If the last letter of B was read and deleted (but B was not emptied), the head of A moves to the first letter of the next block. The while-loop ends as soon as a transition of A is simulated that yields Accept or Reject. It is again easily seen that, for an input of the form $w \cdot \triangleleft$, where $w \in \{a, b\}^n$, the procedure Simulate_nrDFAwtl executes $\mathcal{O}(n)$ many steps.

Finally, we combine these two procedures into an algorithm for solving the membership problem for the nrDFAwtl A (see Algorithm Solve_Membership_Problem_for_nrDFAwtl_A_on_Binary_Alphabet below).

Thus, the membership problem for the nrDFAwtl A is solved in $\mathcal{O}(n)$ steps. As the pointers can be realized by integers of size up to n only, we see that each step just manipulates a finite number of characters or integers of size up to n. As the integers are represented in binary form, each step takes time $\mathcal{O}(\log n)$ only. It follows

Procedure Build_List_of_Blocks:
Input: A word $w \cdot \lhd$, where $w \in \{a, b\}^+$.
Output:A doubly linked list of records of type *block* encoding the word $w \cdot \lhd$.
begin $B_0 := block(\rhd, 1, nil, nil)$; (* B_0 encodes the beginning of the input *)
$x := w[1]$; (* Read the first letter *)
$B_1 := block(x, 1, \uparrow B_0, nil)$;
$B_0.next := \uparrow B_1$; (* $B_0.next$ points to the first block *)
$j := 1$; (* j is the number of the current block *)
$x := w[2]$; (* Read the second letter *)
$i := 2$; (* i is the index of the last letter read *)
while $x \neq \lhd$ **do**
begin
if $x = B_j.symbol$ **then** $B_j.length := B_j.length + 1$
else begin $j := j + 1$;
$B_j := block(x, 1, \uparrow B_{j-1}, nil)$;
$B_{j-1}.next := \uparrow B_j$
end;
$i := i + 1$;
$x := w[i]$; (* Read the next letter *)
end;
$j := j + 1$;
$B_j := block(\lhd, 1, \uparrow B_{j-1}, nil)$; (* The last block encodes the sentinel $\lhd$ *)
$B_{j-1}.next := \uparrow B_j$ (* The sequence of blocks is complete *)
end

that this algorithm can be realized in time $\mathcal{O}(n \cdot \log n)$ by a RAM using the logarithmic cost measure. This completes the proof of Theorem 5.1. □

It remains open whether a corresponding result can also be obtained for the case of larger alphabets. For example, if $\Sigma = \{a, b, c\}$ and $w = (abc)^n$, then we have to skip across i many a- and i many b-factors before we reach the i-th occurrence of the letter c. Thus, if an nrDFAwtl first removes all occurrences of the letter c, each time starting from the left end of its tape, then the above strategy would require in the order of $\mathcal{O}(n^2)$ many steps. Also, if for each factor we store pointers to the last a-, b-, and c-factors preceding it and also to the next a-, b-, and c-factors that follow, then as soon as a factor is erased completely, all pointers referring to it must be updated. Again, this may lead to a quadratic number of steps. However, for the case of alphabets of cardinality larger than two, we have at least the following weaker result.

Theorem 5.2. *The membership problem for an nrDFAwtl over an alphabet of cardinality greater than or equal to* 3 *is decidable in time* $\mathcal{O}(n \cdot (\log n)^2)$.

Proof. Let $A = (Q, \Sigma, \lhd, \tau, q_0, \delta)$ be an nrDFAwtl such that $\Sigma = \{a_1, a_2, \ldots, a_r\}$ for some $r \geq 3$. As in the proof above, we can assume that A never gets into an infinite computation and that it accepts only after reading and deleting its input completely. Moreover, for all states $q \in Q$ and all letters $c \in \Sigma \cup \{\lhd\}$, if $c \notin \tau(p)$ and if $\delta(q, c)$ is undefined, then we extend the transition function δ by taking $\delta(q, c) = \mathsf{Reject}$.

Let $w = x_1 x_2 \cdots x_n \in \Sigma^n$ for some $n \geq 1$. In order to determine whether the word w is accepted by the nrDFAwtl A, we proceed as follows.

For each letter $a \in \Sigma$, a balanced binary search tree (see, *e.g.*, [6]) T_a is constructed so that T_a contains those indices $j \in \{1, 2, \ldots, n\}$ for which $x_j = a$ holds, that is, it contains all those positions within the word w at which the letter a occurs. Furthermore, T_a will contain a node with the label $n + 1$, which refers to the right sentinel $\lhd$. During the simulation of the computation of A on input w, we shall search for the smallest value j

```
Procedure Simulate_nrDFAwtl:
Input: A doubly linked list of records of type block that encode a word w · ◁,
       where w ∈ {a, b}^+.
Output: Accept, if w ∈ L(A), or Reject, if w ∉ L(A).
begin
  q := q_0;            (* q is the current state of A *)
  B := B_0.next↑;      (* B is the first block corresponding to a factor of w *)
  r := 1;              (* r is the head position of A within the current block *)
  while q ∈ Q do
    begin
      if (τ(q) = {a, b}) or (B.symbol = ◁) then
        begin q := δ(q, ◁); B := B_0.next↑; r := 1 end
      else
        if B.symbol ∈ τ(q) then      (* Go to the next block *)
          begin B := B.next↑; r := 1 end
      else
        begin
          q := δ(q, B.symbol);
          B.length := B.length − 1;
          if B.length = 0 then              (* Current block is now empty *)
            if B.prev↑.symbol = ▷ then          (* B is the first block *)
              begin B_0.next := B.next; B := B.next↑; r := 1 end
            else
              if (B.prev↑.symbol ∈ {a, b}) and (B.next↑.symbol ∈ {a, b})
              then          (* B is an inner block *)
                begin
                  C := B.prev↑;
                  D := B.next↑;
                  C.next := D.next;
                  C.length := C.length + D.length;
                  r := C.length + 1;          (* The current block is removed and *)
                  B := C                      (* its neighboring blocks are merged *)
                end
              else (* The current block is the last one *)
                begin B.prev↑.next := B.next;
                      B := B.next↑
                end
            else if r > B.length then
                   begin B := B.next↑; r := 1 end
        end
    end;        (* while-loop ends here *)
  if q = Accept then output(Accept) else output(Reject)
end.
```

in a tree T_a such that j is greater than a given key $i \in \{0, 1, 2, \ldots, n\}$, where the value i itself does not occur in the tree T_a. Thus, the node with the value $n + 1$ ensures that this search will always succeed. As we use balanced binary search trees, each of the operations of inserting a value (and rebalancing the tree), searching for a value, and removing a value (and rebalancing the tree) can be executed in $\mathcal{O}(\log n)$ steps.

Algorithm Solve_Membership_Problem_for_nrDFAwtl_A_on_Binary_Alphabet:
Input: A word $w \cdot \lhd$, where $w \in \{a, b\}^*$.
Output:*Accept*, if $w \in L(A)$, or *Reject*, if $w \notin L(A)$.
begin if $|w| = 0$ **then if** $\lambda \in L(A)$ **then output**$(Accept)$
else output$(Reject)$
else begin $List :=$ Build_List_of_Blocks$(w \cdot \lhd)$;
Simulate_nrDFAwtl$(List)$
end
end.

The algorithm Solve_Membership_Problem_for_nrDFAwtl_A_on_Large_Alphabet given below solves the membership problem for A.

Algorithm Solve_Membership_Problem_for_nrDFAwtl_A_on_Large_Alphabet:
Input: A word $w \cdot \lhd$, where $w \in \Sigma^*$.
Output:*Accept*, if $w \in L(A)$, or *Reject*, if $w \notin L(A)$.
begin for all $a \in \Sigma$ **do** $T_a :=$ empty balanced binary search tree;
$i := 1$; $x := w[1]$;
while $x \neq \lhd$ **do**
begin $insert(i, T_x)$; (∗ Insert i into the tree for the letter x ∗)
$i := i + 1$; $x := w[i]$
end;
for all $a \in \Sigma$ **do** $insert(i, T_a)$; (∗ Insert $n + 1$ into all trees ∗)
(∗ Now the simulation of the nrDFAwtl A on $w \cdot \lhd$ begins ∗)
$p := q_0$; (∗ p is the current state of A ∗)
$s := 0$; (∗ s points before the current head position of A ∗)
repeat for all $a \in (\Sigma \setminus \tau(p))$ **do** $min_a := \min\{\, i \in T_a \mid i > s \,\}$;
(∗ For each letter a that is not translucent for state p, ∗)
(∗ the first occurrence to the right of the current head ∗)
(∗ position s is determined if there is any ∗)
$s := \min(\{\, min_a \mid a \in \Sigma \setminus \tau(p) \,\} \cup \{n + 1\})$;
(∗ s is now the new head position ∗)
$p := \delta(p, w[s])$; (∗ Read the letter at position s or $\lhd$ ∗)
if $s < n + 1$ **then** $delete(s, T_{w[s]})$
(∗ Remove position s from the corresponding tree ∗)
else $s := 0$ (∗ On reading $\lhd$, a new cycle begins ∗)
until $p \notin Q$; (∗ The repeat loop ends when $p \in \{\mathsf{Accept}, \mathsf{Reject}\}$ ∗)
if $p =$ Accept **then output**$(Accept)$ **else output**$(Reject)$
end.

First, for each letter $a \in \Sigma$, a binary search tree T_a is generated that contains the positions of all occurrences of the letter a in the given word w and the value $n + 1$, p is set to the initial state, and s is set to the index before the current head position. Then, for each letter a that is not translucent for the current state, the first position min_a of an occurrence of the letter a is determined from the tree T_a such that min_a is to the right of the position s. If there is no such position, then $min_a = n + 1$ is returned. Finally, the minimum s of all values min_a is chosen, where a is from the set $\Sigma \setminus \tau(p)$ of all letters that are not translucent for state p. Then, s is the position of the next letter that A would read (and delete). In addition, if $\tau(p) = \Sigma$, then $s = n + 1$ is

taken, as in this case, the next letter encountered by A is the right sentinel $\triangleleft$. Next, the transition $\delta(p, w[s])$ of A is simulated, where $w[n+1]$ refers to the sentinel $\triangleleft$, and if $s < n+1$, that is, a letter $a \in \Sigma$ occurs at position s, then this position is deleted from the tree T_a. This process continues until an accepting or rejecting transition of A is reached. Note that the input $w \cdot \triangleleft$ is not modified during the simulation of the computation of the nrDFAwtl A.

It can be easily seen that this algorithm processes the input $w \cdot \triangleleft$ in $\mathcal{O}(n \cdot \log n)$ steps. As each step just manipulates a finite number of characters or integers of size up to $n+1$, it follows that this algorithm can be realized in time $\mathcal{O}(n \cdot (\log n)^2)$ by a RAM using the logarithmic cost measure. □

It remains open whether the upper bound of $\mathcal{O}(n \cdot (\log n)^2)$ for the time complexity of the membership problem for an nrDFAwtl on an alphabet of cardinality larger than two can be improved. For the nondeterministic case, we obtain that the membership problem for an nrNFAwtl can be solved nondeterministically in time $\mathcal{O}(n \cdot \log n)$ in the case of a binary alphabet and in time $\mathcal{O}(n \cdot (\log n)^2)$ in the case of an alphabet of cardinality larger than two. However, we currently do not have any subexponential upper bound for the deterministic time complexity of the fixed membership problem for the nrNFAwtl.

We close this section by taking a look at the emptiness problem and the finiteness problem for non-returning finite automata with translucent letters. These problems can be stated as follows:

The *emptiness problem* for nrNFAwtls:

INSTANCE : An nrNFAwtl A.
QUESTION : Is the language $L(A)$ empty?

The *finiteness problem* for nrNFAwtls:

INSTANCE : An nrNFAwtl A.
QUESTION : Is the language $L(A)$ finite?

For NFAwtls, these problems are decidable in polynomial time. In fact, from an NFAwtl A, one can construct an NFA B such that $L(B)$ is a subset of $L(A)$ that is letter-equivalent to $L(A)$, which means that $L(B)$ is a regular sublanguage of $L(A)$ that has the same Parikh image as $L(A)$. Therefore, the language $L(A)$ is empty (or finite) if and only if the language $L(B)$ is empty (or finite). In particular, this shows that the language $L(A)$ is empty if and only if the transition graph of the NFA B does not contain a path from an initial state to a final state. However, the situation is more complicated for non-returning NFAs with translucent letters.

From the diagram describing a given nrNFAwtl A, we can immediately extract information on the patterns of the words that A can scan during a single sweep (or cycle). Of course, if there is a sweep that starts in an initial state and that reaches a state in which A accepts at the end-of-tape marker, then the corresponding words are accepted by A, which means that they are witnesses for the fact that the language $L(A)$ is non-empty. In general, however, A may not have any accepting computations that just consist of single sweeps. For example, this is the case for the nrDFAwtl A_{ex3} considered in Example 3.4. In this situation, each accepting computation consists of a sequence of sweeps such that the words that A scans during these sweeps form a sequence that can be combined into an accepted word. However, it is not clear whether this can always be done. To illustrate this problem, we consider a simple example.

Example 5.3. Let $A = (Q, \Sigma, \triangleleft, \tau, q_0, \delta)$ be the nrDFAwtl that is obtained by taking $Q = \{q_0, q_1, q_2, q_3, q_4, q_5\}$, $\Sigma = \{a, b, c\}$, and by defining the functions τ and δ as follows:

$$\begin{array}{llllllllll}
\tau(q_0) & = & \{b\}, & \tau(q_2) & = & \{c\}, & \tau(q_4) & = & \{a\}, \\
\tau(q_1) & = & \{c\}, & \tau(q_3) & = & \emptyset, & \tau(q_5) & = & \emptyset, \\
\delta(q_0, a) & = & q_1, & \delta(q_2, b) & = & q_3, & \delta(q_4, c) & = & q_5, \\
\delta(q_1, \triangleleft) & = & q_2, & \delta(q_3, \triangleleft) & = & q_4, & \delta(q_5, \triangleleft) & = & \mathsf{Accept}.
\end{array}$$

This nrDFAwtl is depicted by the diagram in Figure 4.

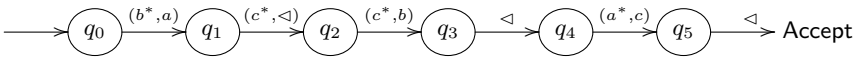


FIGURE 4. The nrDFAwtl A from Example 5.3.

From this diagram, we can immediately extract three types of cycles:

$$\begin{array}{lllllll} (1) & q_0b^*ac^*\cdot\triangleleft & \vdash_A & b^*q_1c^*\cdot\triangleleft & \vdash_A & q_2b^*c^*\cdot\triangleleft, \\ (2) & q_2c^*b\cdot\triangleleft & \vdash_A & c^*q_3\cdot\triangleleft & \vdash_A & q_4c^*\cdot\triangleleft, \\ (3) & q_4a^*c\cdot\triangleleft & \vdash_A & a^*q_5\cdot\triangleleft & \vdash_A & \mathsf{Accept}. \end{array}$$

However, these three cycles cannot be combined into an accepting computation of A. The third cycle (or sweep) requires that there is an occurrence of the letter c, which may only be preceded by occurrences of the letter a. The second cycle requires that there is an occurrence of the letter b, which may only be preceded by occurrences of the letter c. Finally, the first cycle requires that there is an occurrence of the letter a that may only be preceded by occurrences of the letter b and that may only be followed by occurrences of the letter c. Together, these requirements imply that there is no word that A accepts, that is, $L(A) = \emptyset$.

Thus, it remains to determine whether, from a given finite set of patterns of words that are accepted by an nrNFAwtl A in different sweeps (or cycles), one can extract sufficient information for deciding whether there exists a word that is accepted by A, that is, whether, from the various patterns, a word can be obtained that is compatible with all these patterns. Accordingly, it remains open whether the emptiness problem or the finiteness problem are decidable for non-returning NFAs (DFAs) with translucent letters.

6. CONCLUSION

We have extended the NFAwtl and its deterministic variant, the DFAwtl, to the non-returning NFAwtl and the non-returning DFAwtl by abandoning the requirement that, in each step, the automaton reads and deletes the first letter from the beginning of the current word on its tape that is not translucent for the current state. As it turned out, the non-returning types of automata are indeed more expressive than the original types. We then compared the non-returning NFAwtl and the non-returning DFAwtl to some other types of automata that do not read their inputs strictly from left to right. Here, the deterministic and nondeterministic right one-way jumping finite automata ((N)ROWJFA) are of particular interest, as the nrNFAwtl (nrDFAwtl) differs from the NROWJFA (ROWJFA) only in that the former can detect the end of the input, which is marked by the sentinel $\triangleleft$, while the latter has no way of doing so. We have seen that this ability yields a proper increase in expressive capacity. In fact, we obtained a complete hierarchy of language classes that are accepted by the various types of finite automata considered. Also, we derived some closure and non-closure properties for the classes of languages that are accepted by nrNFAwtls and nrDFAwtls. Finally, we established that the membership problem for a non-returning DFAwtl is decidable in time $\mathcal{O}(n \cdot \log n)$ for the case of a binary alphabet and that it is decidable in time $\mathcal{O}(n \cdot (\log n)^2)$ in the general case. However, many questions concerning the nrNFAwtl and the nrDFAwtl are still open:

1. Are all languages accepted by nrNFAwtls necessarily semi-linear? While for unary languages, this is indeed the case, the question remains open for non-unary languages.
2. Is the language class $\mathcal{L}(\mathsf{nrNFAwtl})$ closed under intersection with regular sets? If it is, then we see from Example 3.4 that $\mathcal{L}(\mathsf{nrNFAwtl})$ contains the language L_{ex3}, which is not semi-linear. However, we conjecture that $\mathcal{L}(\mathsf{nrNFAwtl})$ is not closed under this operation, as we expect that the language L_{ex3} is not accepted by any nrNFAwtl.
3. Are the language classes $\mathcal{L}(\mathsf{nrNFAwtl})$ and $\mathcal{L}(\mathsf{nrDFAwtl})$ closed under product, Kleene star, or inverse morphisms?

4. Can the upper bound of $\mathcal{O}(n \cdot (\log n)^2)$ for the time complexity of the membership problem of an nrDFAwtl be improved to $\mathcal{O}(n \cdot \log n)$ also for alphabets of cardinality larger than two?
5. Is emptiness or finiteness decidable for nrDFAwtls or for nrNFAwtls?

Currently, pumping techniques, as used in the proofs of Propositions 3.5 and 4.18, are our only means for proving that a given language is not accepted by any nrNFAwtl. In order to solve the open problems above, it appears to be necessary to develop other techniques for this task. Finally, in [14], simple picture-to-string transducers are used in combination with DFAs and with DFAwtls to define classes of two-dimensional picture languages. Which picture languages are obtained when combining the picture-to-string transducers of [14] with the non-returning DFAwtl?

Acknowledgements. Some of the results of this paper were announced at NCMA 2022 in Debrecen, Hungary, in August 2022. An extended abstract can be found in the proceedings of that conference [13].

References

[1] S. Beier and M. Holzer, Properties of right one-way jumping finite automata. *Theoret. Comput. Sci.* **798** (2019) 78–94.
[2] S. Beier and M. Holzer, Nondeterministic right one-way jumping finite automata. *Inform. Comput.* **284** (2022) 104687.
[3] S. Bensch, H. Bordihn, M. Holzer and M. Kutrib, On input-revolving deterministic and nondeterministic finite automata. *Inform. Comput.* **207** (2009) 1140–1155.
[4] G. Buntrock, F. Otto, Growing context-sensitive languages and Church–Rosser languages. *Inform. Comput.* **141** (1998) 1–36.
[5] H. Chigahara, S.Z. Fazekas and A. Yamamura, One-way jumping finite automata. *Int. J. Found. Comput. Sci.* **27** (2016) 391–405.
[6] T.H. Cormen, C.E. Leiserson, R.L. Rivest and C. Stein, Introduction to Algorithms, 4th edn. MIT Press (2022).
[7] V. Diekert and G. Rozenberg, The Book of Traces. World Scientific, Singapore (1995).
[8] H. Fernau, M. Paramasivan and M.L. Schmid, Jumping finite automata: characterizations and complexity, edited by F. Drewes. *CIAA 2012, Proc.*, Lecture Notes in Computer Science 9223. Springer, Heidelberg (2012) 89–101.
[9] P. Jančar, F. Mráz, M. Plátek and J. Vogel, Restarting automata, in edited by H. Reichel. *FCT'95, Proc.*, Lecture Notes in Computer Science 965. Springer, Berlin (1995) 283–292.
[10] R. Loukanova, Linear context free languages, in edited by C.B. Jones, Z. Liu and J. Woodcock. *ICTAC 2007, Proc.*, Lecture Notes in Computer Science 4711. Springer, Heidelberg (2007) 351–365.
[11] A. Meduna and P. Zemek, Jumping finite automata. *Int. J. Found. Comput. Sci.* **23** (2012) 1555–1578.
[12] F. Mráz, Lookahead hierarchies of restarting automata. *J. Automata Lang. Combinatorics* **6** (2001) 493–506.
[13] F. Mráz and F. Otto, Non-returning finite automata with translucent letters, in edited by H. Bordihn, G. Horváth and G. Vaszil. 12th International Workshop on Non-Classical Models of Automata and Applications (NCMA 2022). Vol. 367 of *EPTCS*. Open Publishing Association (2022) 143–159.
[14] F. Mráz and F. Otto, Recognizing picture languages by reductions to string languages. *J. Automata Lang. Combinatorics* **27** (2022) 199–228.
[15] B. Nagy, On $5' \to 3'$ sensing Watson–Crick automata, in edited by M. Garzon and H. Yan. DNA Computing, 13th Intern. Meeting, Revised Selected Papers, Lecture Notes in Computer Science 4848. Springer, Heidelberg (2008) 256–262.
[16] B. Nagy and L. Kovács, Finite automata with translucent letters applied in natural and formal language theory, in edited by N.T. Nguyen, R. Kowalczyk, A. Fred and F. Joaquim. *Transactions on Computational Collective Intelligence XVII*, Lecture Notes in Computer Science 8790. Springer, Heidelberg (2014) 107–127.
[17] B. Nagy and F. Otto, CD-systems of stateless deterministic R(1)-automata accept all rational trace languages, in edited by A.H. Dediu, H. Fernau and C. Martin-Vide. *LATA 2010, Proc.*, Lecture Notes in Computer Science 6031. Springer, Berlin (2010) 463–474.
[18] B. Nagy and F. Otto, Finite-state acceptors with translucent letters, in edited by G. Bel-Enguix, V. Dahl and A.O. De La Puente. *BILC 2011: AI Methods for Interdisciplinary Research in Language and Biology, Proc.* SciTePress, Portugal (2011) 3–13.
[19] B. Nagy and F. Otto, Globally deterministic CD-systems of stateless R(1)-automata, in edited by A.H. Dediu, S. Inenaga and C. Martín-Vide. *Language and Automata Theory and Applications, LATA 2011, Proc.*, Lecture Notes in Computer Science 6638. Springer, Berlin (2011) 390–401.
[20] B. Nagy and F. Otto, On CD-systems of stateless deterministic R-automata with window size one. *J. Comput. Syst. Sci.* **78** (2012) 780–806.
[21] B. Nagy and F. Otto, Globally deterministic CD-systems of stateless R-automata with window size 1. *Int. J. Comput. Math.* **90** (2013) 1254–1277.
[22] F. Otto, Restarting automata, in edited by Z. Ésik, C. Martín-Vide and V. Mitrana. Recent Advances in Formal Languages and Applications. Vol. 25 of *Studies in Computational Intelligence*. Springer, Heidelberg (2006) 269–303.