Non-Returning Finite Automata With Translucent Letters

František Mráz
Charles University
Department of Computer Science
Malostranské nám. 25
118 00 PRAHA, Czech Republic
frantisek.mraz@mff.cuni.cz

Friedrich Otto
Universität Kassel
Fachbereich Elektrotechnik/Informatik
34109 KASSEL, Germany
f.otto@uni-kassel.de

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 rather continues from the position of the letter just deleted. When the end-of-tape marker is reached, our automaton can decide whether to accept, to reject, or to continue, which means that it again reads the remaining tape contents 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. Also we are interested in closure properties of the resulting classes of languages and in decision problems.

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. Under this aspect, the most extreme is the *jumping finite automaton* of Meduna and Zemek [7] (see also [4]), 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 [6], 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 alternatingly the first and the last letter. 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 [9].

Moreover, there is the *restarting automaton* as introduced by Jančar, Mráz, Plátek, and Vogel in [5], which processes a given input in cycles, in each cycle scanning the remaining input from left to right until it deletes one or more letters, returns its head to the left end of the remaining input, and reenters its initial state. If using a head 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., [16]), while with a head of size one, they accept exactly the regular languages [8].

Finally, there is the (deterministic and nondeterministic) finite automaton *with translucent letters* (or DFAwtl and NFAwtl) of Nagy and Otto [12], which is equivalent to a cooperating distributed system of stateless deterministic R-automata with heads 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 that contains 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 that it can see, that is, that is not translucent for the 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 [11, 13, 14, 15]. In addition, while the obvious upper bound for the time complexity of the membership problem for a DFAwtl is DTIME(n^2), a better upper bound of DTIME($n \cdot \log n$) is derived in [10].

Here we propose a variant of the nondeterministic finite automaton with translucent letters which, after reading and deleting a letter, does not return to the left end of its tape, but that rather 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 again reads the remaining tape contents 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 membership problems, but we are also interested in 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, we explain its workings by a detailed example, and we derive a kind of normalized form for this type of automaton. In the next section, which is the main part of the paper, we compare the classes of languages that are 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, to the rational trace languages, and to the classes of the Chomsky hierarchy, establishing some proper inclusion results and some incomparability results. Then, in Section 4, 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. Finally, in Section 5, we study the complexity of the membership problem for the nrDFAwtl, showing that it is decidable in time $O(n \cdot (\log n)^2)$ whether a word of length n is accepted by a given nrDFAwtl. In the concluding section, we summarize our results and state a number of open problems for future work.

2 Definitions

In order to use it as a reference, we restate the definition of the nondeterministic finite automaton with translucent letters from [12].

Definition 1 *A* finite automaton with translucent letters*, an* NFAwtl *for short, is defined as a 7-tuple $A = (Q, \Sigma, \lhd, \tau, I, F, \delta)$, 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*, $\tau : Q \to \mathscr{P}(\Sigma)$ is a* translucency mapping*, $I \subseteq Q$ is a set of initial states, $F \subseteq Q$ is a set of final states, and $\delta : Q \times \Sigma \to \mathscr{P}(Q)$ 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 $u \in (\tau(q))^*$

and $a \notin \tau(q)$, 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 reaches 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:

$$qw \cdot \lhd \vdash_A \begin{cases} q'uv \cdot \lhd, & \text{if } w = uav,\ u \in (\tau(q))^*,\ a \notin \tau(q), \text{ and } q' \in \delta(q,a), \\ \mathsf{Reject}, & \text{if } w = uav,\ u \in (\tau(q))^*,\ a \notin \tau(q_0), \text{ and } \delta(q,a) = \emptyset, \\ \mathsf{Accept}, & \text{if } w \in (\tau(q))^* \text{ and } q \in F, \\ \mathsf{Reject}, & \text{if } w \in (\tau(q))^* \text{ and } q \notin F. \end{cases}$$

A word $w \in \Sigma^*$ is *accepted by A* if there exists an initial state $q_0 \in I$ and a computation $q_0 w \cdot \lhd \vdash_A^* \mathsf{Accept}$, where $\vdash_A^*$ denotes the reflexive transitive closure of the above 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 $\mathscr{L}(\mathsf{NFAwtl})$ denotes the class of all languages that are accepted by NFAwtls.

Definition 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 $\mathscr{L}(\mathsf{DFAwtl})$ denotes the class of all languages that are accepted by DFAwtls.*

For future reference, we present an example of a DFAwtl.

Example 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:*

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

and $\delta(q_a,b), \delta(q_b,a), \delta(q_a',b')$, and $\delta(q_b',a')$ are undefined. For the word $abba'b'ab'a'$, A_c executes the following accepting computation:

$$\begin{array}{llllllll} q_0abba'b'ab'a'\lhd & \vdash_{A_c} & q_a'bba'b'ab'a'\lhd & \vdash_{A_c} & q_0bbb'ab'a'\lhd & \vdash_{A_c} & q_b'bb'ab'a'\lhd \\ & \vdash_{A_c} & q_0bab'a'\lhd & \vdash_{A_c} & q_b'ab'a'\lhd & \vdash_{A_c} & q_0aa'\lhd \\ & \vdash_{A_c} & q_a'a'\lhd & \vdash_{A_c} & q_0\lhd & \vdash_{A_c} & \mathsf{Accept}. \end{array}$$

In fact, if φ denotes the morphism that is defined through $\varphi(a) = a'$ and $\varphi(b) = b'$, then it is easily checked that $L(A_c) = \{\, \mathrm{sh}(w, \varphi(w)) \mid w \in \{a,b\}^* \,\}$, where sh *denotes the shuffle operation.* ■

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 [2]. 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 3 shows that already DFAwtls accept quite some complicated languages in comparison to the Chomsky hierarchy. On the other hand, the language $L(A_c)$ is the rational trace language that is obtained from the regular language $\{aa', bb'\}^*$ through the dependency relation $D = \{(a,b),(b,a),(a',b'),(b',a')\}$ (see, e.g., [11, 14]).

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 for the current state, deletes it, changes its state, and returns to the first letter. Here we propose a variant of this type of automaton that does not necessarily return to the first letter, but that continues from the position of the letter deleted, returning to the first letter only after the tape contents has been scanned 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 4 *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, *$\tau : Q \to \mathscr{P}(\Sigma)$ is a* translucency mapping, *$I \subseteq Q$ is a set of initial states, and*

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

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.*

From the above definition, we see that $\delta(q, \lhd)$ is either a subset of Q or the operation Accept, that is, on 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 $xq_1 w \cdot \lhd$, 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 $u \in (\tau(q_1))^*$ and $a \notin \tau(q_1)$, 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 $xuq_2 v \cdot \lhd$. 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 $q_2 xw \cdot \lhd$. 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:

$$xqw \cdot \lhd \vdash_A \begin{cases} xuq'v \cdot \lhd, & \text{if } w = uav,\, u \in (\tau(q))^*,\, a \notin \tau(q), \text{ and } q' \in \delta(q,a), \\ \mathsf{Reject}, & \text{if } w = uav,\, u \in (\tau(q))^*,\, a \notin \tau(q_0), \text{ and } \delta(q,a) = \emptyset, \\ q'xw \cdot \lhd & \text{if } w \in (\tau(q))^* \text{ and } q' \in \delta(q, \lhd), \\ \mathsf{Accept}, & \text{if } w \in (\tau(q))^* \text{ and } \delta(q, \lhd) = \mathsf{Accept}, \\ \mathsf{Reject}, & \text{if } w \in (\tau(q))^* \text{ and } \delta(q, \lhd) = \emptyset. \end{cases}$$

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

Definition 5 *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 $q_1 wu \cdot \lhd \vdash_A^* w' q_2 u \cdot \lhd$, 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*

$$q_1 wu \cdot \lhd \vdash_A^s w' u q_2 \lhd$$

to denote the above sweep. Observe that the configurations $w' q_2 u \cdot \lhd$ and $w' u q_2 \lhd$ 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 $q_1 wu \cdot \lhd \vdash_A^s w' u q_2 \lhd$ together with the next transitional step $q_3 \in \delta(q_2, \lhd)$. Thus, a cycle has the form $q_1 wu \cdot \lhd \vdash_A^* w' q_2 u \cdot \lhd \vdash_A q_3 w' u \cdot \lhd$. We use the notation*

$$q_1 wu \cdot \lhd \vdash_A^c q_3 w' u \cdot \lhd$$

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 $q_0 w \cdot \lhd \vdash_A^* \mathsf{Accept}$. Then

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

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

Definition 6 *An nrNFAwtl $A = (Q, \Sigma, \lhd, \tau, I, \delta)$ is a* non-returning deterministic finite-state acceptor with translucent letters, *abbreviated as* nrDFAwtl, *if $|I| = 1$ and if $|\delta(q, a)| \leq 1$ for all $q \in Q$ and all $a \in \Sigma \cup \{\lhd\}$. Then $\mathscr{L}(\mathsf{nrDFAwtl})$ denotes the class of all languages that are accepted by nrDFAwtls.*

We illustrate these definitions by an example.

Example 7 *Let $A = (Q, \{a, b, c\}, \lhd, \tau, \{q_a\}, \delta)$ be the nrDFAwtl 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 executes the following accepting computation:*

$$\begin{array}{llllll} q_a aabbcc \cdot \lhd & \vdash_A & q_b abbcc \cdot \lhd & \vdash_A & aq_c bcc \cdot \lhd & \vdash_A & abq_r c \cdot \lhd \\ & \vdash_A & q_a abc \cdot \lhd & \vdash_A & q_b bc \cdot \lhd & \vdash_A & q_c c \cdot \lhd \\ & \vdash_A & q_r \lhd & \vdash_A & q_a \lhd & \vdash_A & \mathsf{Accept}, \end{array}$$

that is, A accepts on input $w = aabbcc$. In fact, $q_a aabbcc \cdot \lhd \vdash_A^s abcq_r \lhd$ is a sweep and $q_a aabbcc \cdot \lhd \vdash_A^c q_a abc \cdot \lhd$ is a cycle of A. Actually, it is easily seen that $L(A) = \{ a^n b^n c^n \mid n \geq 0 \}$. ■

Recall from [12] that the language $\{ a^n b^n c^n \mid n \geq 0 \}$ is not accepted by any NFAwtl.

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 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. ■

In general, 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 8 as $\tau(p) = \tau(r) = \{a\}$, $r \in \delta(p,\lhd)$, and $\delta(r,\lhd) = \mathsf{Accept}$. However, we can easily convert 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:

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

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 follows easily that $L(C) = L(A)$. Together the two constructions considered yield the following technical result.

Proposition 9 *Each nrNFAwtl A can effectively be converted into an equivalent nrNFAwtl C that never gets into an infinite computation and that accepts only after reading and deleting its tape contents completely. In addition, if A is deterministic, then so is C.*

3 Proper Inclusion Results and Incomparability Results

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

Theorem 10 *From a given NFAwtl A, one can 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 a simulating nrNFAwtl $B = (Q_B,\Sigma,\lhd,\tau_B,I_B,\delta_B)$ 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 $\delta_B(q',a) = \emptyset$. Moreover, $\delta_B(q,\lhd) = \mathsf{Accept}$, if $q \in F$, and $\delta_B(q',\lhd) = \{q\}$.

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 two cases that we must consider.

- First assume that $w = uav$ for some word $u \in (\tau(q))^*$ and a letter $a \notin \tau(q)$. If $p \in \delta(q,a)$, then $qw \cdot \lhd \vdash_A puv \cdot \lhd$ is a possible step of A. In this case, B can execute the following sequence of steps:
$$qw \cdot \lhd = quav \cdot \lhd \vdash_B up'v \cdot \lhd \vdash_B puv \cdot \lhd.$$
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.
- If $w \in (\tau(q))^*$, then A accepts, if $q \in F$, otherwise, it rejects. In this case, B just acts likewise.

Thus, 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. It follows 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 10. □

Together with Example 7, this theorem has the following consequence.

Corollary 11 $\mathscr{L}(\mathsf{NFAwtl}) \subsetneq \mathscr{L}(\mathsf{nrNFAwtl})$ *and* $\mathscr{L}(\mathsf{DFAwtl}) \subsetneq \mathscr{L}(\mathsf{nrDFAwtl})$.

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 m is the cardinality of the underlying alphabet. Does a corresponding result also hold for nrNFAwtls? First we consider this question for the special case of a unary alphabet.

Proposition 12 *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 accepted by an NFA and therewith also by an NFAwtl. Theorem 10 then shows that L is accepted by an nrNFAwtl.

Conversely, assume that a language $L \subseteq \{a\}^*$ is accepted by an nrNFAwtl $A = (Q,\{a\},\lhd,\tau,I,\delta)$. By Proposition 9, we can assume that the nrNFAwtl A never gets into an infinite computation and that it accepts only after reading and deleting its tape contents 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:

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

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 $\delta(q,a)$ is undefined. Hence,

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

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

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

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) a 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 12. $\square$

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 13 *We define the nrDFAwtl* $A_{\text{ex3}} = (Q, \Sigma, \lhd, \tau, I, \delta)$ *as follows:*

- $Q = \{q_0, q_1, q_2, q_3, q_4, q_5, q_6, q_7, q_8\}$, $\Sigma = \{a,b,c\}$, *and* $I = \{q_0\}$,
- $$\begin{array}{lcllclcl} \tau(q_0) & = & \{a\}, & \tau(q_1) & = & \tau(q_2) & = & \emptyset, \\ \tau(q_3) & = & \{b\}, & \tau(q_4) & = & \tau(q_5) & = & \emptyset, \\ \tau(q_6) & = & \{a,c\}, & \tau(q_7) & = & \tau(q_8) & = & \emptyset, \end{array}$$
- *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* $q_i \xrightarrow{x} q_j$ *denotes a transition from* q_i *to* q_j *that simply reads an occurrence of the letter x, and an edge of the form* $q_i \xrightarrow{(Y^*,x)} q_j$ *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 a restart or an accept operation.*

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

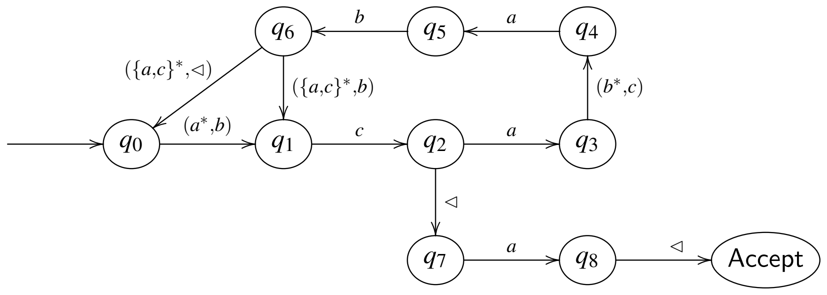

Figure 1: The diagram describing the nrDFAwtl A_{ex3}

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 on to q_8. Thus, during this part, a single occurrence of each of the letters a, b, and c is removed.*

Together these observations imply that during each accepting computation, A_{ex3} removes the same uneven 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 $n \geq 1$. This input yields the following computation:

$$\begin{array}{rlll}
q_0(abc)^{3n}\cdot\triangleleft & = q_0abc(abc)^{3n-1}\cdot\triangleleft & \vdash_{A_{\text{ex3}}} aq_1c(abc)^{3n-1}\cdot\triangleleft & \vdash_{A_{\text{ex3}}} aq_2abc(abc)^{3n-2}\cdot\triangleleft\\
& \vdash_{A_{\text{ex3}}} aq_3bc(abc)^{3n-2}\cdot\triangleleft & \vdash_{A_{\text{ex3}}} abq_4abc(abc)^{3n-3}\cdot\triangleleft & \vdash_{A_{\text{ex3}}} abq_5bc(abc)^{3n-3}\cdot\triangleleft\\
& \vdash_{A_{\text{ex3}}} abq_6cabc(abc)^{3n-4}\cdot\triangleleft & \vdash_{A_{\text{ex3}}} abcaq_1c(abc)^{3n-4}\cdot\triangleleft & \vdash_{A_{\text{ex3}}} abcaq_2abc(abc)^{3n-5}\cdot\triangleleft\\
& \vdash^*_{A_{\text{ex3}}} (abc)^{n-1}aq_2abc(abc)\cdot\triangleleft & \vdash_{A_{\text{ex3}}} (abc)^{n-1}aq_3bc(abc)\cdot\triangleleft & \vdash_{A_{\text{ex3}}} (abc)^{n-1}abq_4abc\cdot\triangleleft\\
& \vdash_{A_{\text{ex3}}} (abc)^{n-1}abq_5bc\cdot\triangleleft & \vdash_{A_{\text{ex3}}} (abc)^{n-1}abq_6c\cdot\triangleleft & \vdash_{A_{\text{ex3}}} q_0(abc)^{n-1}abc\cdot\triangleleft\\
& = q_0(abc)^n\cdot\triangleleft. & &
\end{array}$$

Finally, as

$$q_0abc\cdot\triangleleft \vdash_{A_{\text{ex3}}} aq_1c\cdot\triangleleft \vdash_{A_{\text{ex3}}} aq_2\triangleleft \vdash_{A_{\text{ex3}}} q_7a\cdot\triangleleft \vdash_{A_{\text{ex3}}} q_8\triangleleft \vdash_{A_{\text{ex3}}} \mathsf{Accept},$$

it follows that $L_{\text{exp3}} = \{\,(abc)^{3^n} \mid n \geq 0\,\} \subseteq L(A_{\text{ex3}})$. Unfortunately, A_{ex3} also accepts some words that do not belong to the language L_{exp3}. In fact, it can be shown that $L' = \{\,ab(cacabb)^nc \mid n \geq 0\,\} \subseteq L(A_{\text{ex3}})$. Indeed, for $n = 0$, we have $ab(cacabb)^nc = abc \in L(A_{\text{ex3}})$. Now, proceeding by induction on n,

$$\begin{array}{rcllll}
q_0ab(cacabb)^{n+1}c\cdot\triangleleft & = & q_0abcacabb(cacabb)^nc\cdot\triangleleft & \vdash_{A_{\text{ex3}}} & aq_1cacabb(cacabb)^nc\cdot\triangleleft\\
& \vdash^6_{A_{\text{ex3}}} & aq_1(cacabb)^nc\cdot\triangleleft & \vdash^*_{A_{\text{ex3}}} & \mathsf{Accept}.
\end{array}$$

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

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

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

On the other hand, we have the following fact.

Claim. $L(A_{\text{ex3}}) \cap (abc)^* = L_{\text{exp3}}$.

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

$$q_0 w \cdot \lhd \vdash^*_{A_{\text{ex3}}} (abc)^{n-1} abq_6 cabc \cdot \lhd \vdash_{A_{\text{ex3}}} (abc)^n aq_1 c \cdot \lhd \vdash_{A_{\text{ex3}}} (abc)^n aq_2 \lhd \vdash_{A_{\text{ex3}}} q_7 (abc)^n a \cdot \lhd,$$

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

$$\begin{array}{llllll} q_0 w \cdot \lhd & \vdash_{A_{\text{ex3}}} & (abc)^n aq_1 cabc \cdot \lhd & \vdash_{A_{\text{ex3}}} & (abc)^n aq_2 abc \cdot \lhd & \vdash_{A_{\text{ex3}}} \; (abc)^n aq_3 bc \cdot \lhd \\ & \vdash_{A_{\text{ex3}}} & (abc)^n abq_4 \lhd & \vdash_{A_{\text{ex3}}} & \mathsf{Reject}. & \end{array}$$

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

Hence, our example shows that the intersection of a language that is accepted by an nrDFAwtl and a regular set is not necessarily semi-linear. ■

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

As all rational trace languages are accepted by NFAwtls, Corollary 11 implies that all rational trace languages are accepted by nrNFAwtls. However, as shown in [15], 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 14 $L_{\vee} \notin \mathscr{L}(\mathsf{nrDFAwtl})$.

Proof. We prove this result by contradiction. So 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 $q_i a^r b^s \cdot \lhd \vdash^s_A a^{r-r_1} b^{s-s_1} q_j \lhd$ 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 copies of the letter a and then it reads (and deletes) s_1 copies 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

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

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 $s_1 \leq m$. □

Now let $n > 3m^2$. Then $a^nb^n \in L_\vee$, and the computation of A on input a^nb^n is accepting. It consists of a sequence of sweeps and an accept step, that is, we have

$$\begin{array}{llcll} q_0a^nb^n \cdot \lhd & \vdash_A^s & a^{n-r_1}b^{n-s_1}q_{i_1} \cdot \lhd & \vdash_A & q_{j_1}a^{n-r_1}b^{n-s_1} \cdot \lhd \\ & \vdash_A^s & a^{n-r_1-r_2}b^{n-s_1-s_2}q_{i_2} \cdot \lhd & \vdash_A & q_{j_2}a^{n-r_1-r_2}b^{n-s_1-s_2} \cdot \lhd \\ & \vdash_A^s & \dots & \vdash_A^s & a^{n-r_1-r_2-\dots-r_k}b^{n-s_1-s_2-\dots-s_k}q_{i_k} \cdot \lhd \\ & \vdash_A & \mathsf{Accept}, & & \end{array}$$

where $k \geq 1$ and $r_i, s_i \leq m$ for all $i = 1,2,\dots,k$. If $n > r_1 + r_2 + \dots + r_k$, then A would also accept the word $a^{n+1}b^n \notin L_\vee$, and if $n > s_1 + s_2 + \dots + s_k$, then A would also accept the word $a^nb^{n+1} \notin L_\vee$. It follows that $n = r_1 + r_2 + \dots + r_k = s_1 + s_2 + \dots + s_k$, that is, A erases its input a^nb^n completely before it accepts. Because of the above claim, this means in particular that the number of sweeps k in the above computation satisfies the inequality $k > 3m$.

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. Hence, the above computation can be written as follows:

$$\begin{array}{lll} q_0a^nb^n \cdot \lhd & \vdash_A^* & q_{j_\alpha}a^{n-r_1-r_2-\dots-r_\alpha}b^{n-s_1-s_2-\dots-s_\alpha} \cdot \lhd \\ & \vdash_A^* & q_{j_\beta}a^{n-r_1-r_2-\dots-r_\alpha-r_{\alpha+1}-\dots-r_\beta}b^{n-s_1-s_2-\dots-s_\alpha-s_{\alpha+1}-\dots-s_\beta} \cdot \lhd \\ & = & q_{j_\alpha}a^{n-r_1-r_2-\dots-r_\alpha-r_{\alpha+1}-\dots-r_\beta}b^{n-s_1-s_2-\dots-s_\alpha-s_{\alpha+1}-\dots-s_\beta} \cdot \lhd \\ & \vdash_A^* & q_{i_k}\lhd \vdash_A \mathsf{Accept}. \end{array}$$

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

$$q_0a^{n+c}b^{n+c'} \cdot \lhd \vdash_A^* q_{j_\alpha}a^{n_\alpha+c}b^{n'_\alpha+c'} \cdot \lhd \vdash_A^* q_{j_\alpha}a^{n_\alpha-c+c}b^{n'_\alpha-c'+c'} \cdot \lhd = q_{j_\alpha}a^{n_\alpha}b^{n'_\alpha} \cdot \lhd \vdash_A^* \mathsf{Accept}.$$

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 $a^nb^{2n} \in L_\vee$. As A is deterministic, this computation looks as follows:

$$q_0a^nb^{2n} \cdot \lhd \vdash_A^* q_{j_\alpha}a^{n_\alpha}b^{n+n'_\alpha} \cdot \lhd \vdash_A^* q_{j_\alpha}a^{n_\alpha-c}b^{n+n'_\alpha-c} \cdot \lhd \vdash_A^* \mathsf{Accept}.$$

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

$$q_0a^{n+c}b^{2n+c} \cdot \lhd \vdash_A^* q_{j_\alpha}a^{n_\alpha+c}b^{n+n'_\alpha+c} \cdot \lhd \vdash_A^* q_{j_\alpha}a^{n_\alpha-c+c}b^{n+n'_\alpha-c+c} \cdot \lhd = q_{j_\alpha}a^{n_\alpha}b^{n+n'_\alpha} \cdot \lhd \vdash_A^* \mathsf{Accept}.$$

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

It thus follows that the class of rational trace languages is not contained in the language class $\mathscr{L}(\mathsf{nrDFAwtl})$. Finally, we consider the subset P_2 of the semi-Dyck language D_2 that is defined by the context-free grammar

$$G = (\{S\}, \{a,b,c,d\}, S, \{(S \to \lambda), (S \to aSc), (S \to bSd)\}.$$

Thus, P_2 consists of all fully bracketed expressions over $\Sigma = \{a,b,c,d\}$, where the letters a and b are seen as opening brackets, the letters c and d are the corresponding closing brackets, and all opening brackets come before all closing brackets.. Hence, P_2 consists of all words of the form

$$w = a^{i_1}b^{j_1}a^{i_2}b^{j_2}\cdots a^{i_k}b^{j_k}d^{j_k}c^{i_k}\cdots d^{j_2}c^{i_2}d^{j_1}c^{i_1},$$

where $k \geq 0$, $i_1, j_k \geq 0$, $i_2, i_3, \ldots, i_k > 0$, and $j_1, j_2, \ldots, j_{k-1} > 0$. It is easily seen that P_2 is a deterministic linear language, that is, it is accepted by a deterministic one-turn pushdown automaton (see, e.g., [1]). For this language we have the following negative result.

Theorem 15 $P_2 \notin \mathscr{L}(\mathsf{nrNFAwtl})$.

Proof outline. We prove this result by contradiction. So assume that $A = (Q, \Sigma, \lhd, \tau, I, \delta)$ is an nrNFAwtl such that $L(A) = P_2$, where $Q = \{q_0, q_1, \ldots, q_{m-1}\}$ and $\Sigma = \{a, b, c, d\}$.

Let $w \in \Sigma^*$ be an input word. A factor of w of maximum length that only consists of occurrences of the letter a is called an *a-block* of w, and analogously, we have *b-blocks*, *c-blocks*, and *d-blocks*. As observed above, w consists of an alternating sequence of a- and b-blocks that is followed by an alternating sequence of c- and d-blocks. It is important to notice that the language P_2 contains words that consist of arbitrarily many blocks of arbitrary size. Based on this observation the following technical results concerning accepting computations of A on inputs of sufficient size can be derived:

(1) During a sweep within an accepting computation of A on an input w, at most m letters can be deleted from any block of size larger than m.

(2) During a sweep within an accepting computation on an input w, A deletes letters from adjacent blocks from the $\{a, b\}^*$-prefix of w starting with the first a-block or the first b-block, and analogously, it deletes letters from adjacent blocks of the $\{c, d\}^*$-suffix, starting with the first d-block or the first c-block.

(3) Within an accepting computation of A, the number of blocks that are modified within a given sweep is bounded from above by a fixed multiple of the number of sweeps already executed.

(4) For any word of the form $w = a^{i_1} b^{j_1} \cdots a^{i_k} b^{j_k} d^{j_k} c^{i_k} \cdots d^{j_1} c^{i_1} \in P_2$, where $k > 6m^2 + 2$ and $i_\nu, j_\nu > m^2$ for all $\nu = 1, 2, \ldots, k$, any accepting computation of A on input w consists of at most m cycles.

These statements are proved in a similar way as Proposition 14. However, more involved arguments based on pumping are needed. Based on these statements, we can now complete the proof as follows.

If $w \in P_2$ is a word of the form described in (4), then an accepting computation of A on input w consists of at most m cycles. During each cycle, at most m letters are deleted from at most $4m \cdot (m+1) = 4m^2 + 4m$ blocks, and so, the word obtained through these cycles still consists of $4k > 24m^2 + 8$ non-empty blocks. As A now accepts, it is obvious that together with the word w, A also accepts words that do not belong to the language P_2. This contradiction shows that there is no nrNFAwtl A such that $L(A) = P_2$. This completes the proof of Theorem 15. $\square$

Thus, the nrNFAwtl does not even accept all deterministic linear languages. The diagram in Figure 2 summarizes the relationships between the classes of languages that are accepted by the various types of finite automata with translucent letters and the classes of the Chomsky hierarchy.

4 Closure and Non-Closure Properties

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

Theorem 16 *The language class* $\mathscr{L}(\mathsf{nrNFAwtl})$ *is closed under union and disjoint shuffle.*

$$\begin{tikzcd}
\mathsf{CSL} & & \\
\mathsf{GCSL} \arrow[u] & \mathscr{L}(\mathsf{nrNFAwtl}) \arrow[ul] & \\
\mathsf{CFL} \arrow[u] & & \mathscr{L}(\mathsf{nrDFAwtl}) \arrow[ul] \\
\mathsf{LIN} \arrow[u] & \mathscr{L}(\mathsf{NFAwtl}) \arrow[uu] & \\
\mathsf{DLIN} \arrow[u] & \mathsf{LRAT} \arrow[u] & \mathscr{L}(\mathsf{DFAwtl}) \arrow[ul] \arrow[uu] \\
\mathsf{REG} \arrow[u] \arrow[ur] \arrow[r, equal] & \mathscr{L}(\mathsf{NFA}) \arrow[r, equal] & \mathscr{L}(\mathsf{DFA}) \arrow[u]
\end{tikzcd}$$

Figure 2: Hierarchy of language classes accepted by the various types of finite automata with translucent letters. Here LRAT denotes the class of all rational trace languages. Each arrow represents a proper inclusion, and classes that are not connected by a sequence of arrows are incomparable under inclusion.

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 two nrNFAwtls. Without loss of generality we may assume that the sets 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

$$\tau(q) = \left\{ \begin{array}{ll} \tau_1(q), & \text{if } q \in Q_1 \\ \tau_2(q), & \text{if } q \in Q_2 \end{array} \right\} \text{ and } \delta(q,a) = \left\{ \begin{array}{ll} \delta_1(q,a), & \text{if } q \in Q_1 \\ \delta_2(q,a), & \text{if } q \in Q_2 \end{array} \right\}$$

for all $a \in \Sigma \cup \{\lhd\}$. Then $L(A) = L(A_1) \cup L(A_2)$, which proves that the class $\mathscr{L}(\mathsf{nrNFAwtl})$ 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 $\mathrm{sh}(L_1, L_2)$ by taking

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

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

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 $\delta_1(q,\lhd) = \mathsf{Accept}$, 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 $\mathscr{L}(\mathsf{nrNFAwtl})$ is closed under disjoint shuffle. $\square$

For the nrDFAwtl, we have the following results.

Theorem 17 *The language class* $\mathscr{L}(\mathsf{nrDFAwtl})$ *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 14. This shows that the class $\mathscr{L}(\mathsf{nrDFAwtl})$ is not closed under union.

Next we prove that the class $\mathscr{L}(\mathsf{nrDFAwtl})$ is closed under complementation. Let $A = (Q, \Sigma, \lhd, \tau, I, \delta)$ be an nrDFAwtl. We define an nrDFAwtl $A^c = (Q \cup \{q_+\}, \Sigma, \lhd, \tau^c, I, \delta^c)$, where q_+ is a new state, by taking

$$\tau^c(q) = \left\{ \begin{array}{ll} \tau(q), & \text{if } q \in Q \\ \Sigma, & \text{if } q = q_+ \end{array} \right\},$$

and by defining, for all $q \in Q$ and all $a \in \Sigma \cup \{\lhd\}$,

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

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 $L(A^c) = \Sigma^* \smallsetminus L(A)$, which shows that the class $\mathscr{L}(\mathsf{nrDFAwtl})$ is closed under complementation.

Closure under complementation and non-closure under union imply that the class $\mathscr{L}(\mathsf{nrDFAwtl})$ 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 14. This proves that the class $\mathscr{L}(\mathsf{nrDFAwtl})$ is not closed under alphabetic morphisms. □

5 Decision Problems

The membership problem for each nrNFAwtl is solvable in linear space. Moreover, it is straightforward to see that $\mathscr{L}(\mathsf{nrNFAwtl}) \subseteq \mathsf{NTIME}(n^2)$. In particular, the membership problem for a nrDFAwtl is decidable in quadratic time. However, by associating, for each letter $a \in \Sigma$, a balanced binary search tree (see, e.g., [3]) T_a to a word of length n over Σ such that T_a contains those indices $i \in \{1,2,\ldots,|w|\}$ at which the letter a occurs in the word w, it can be shown that the computation of a nrDFAwtl A on a word of length n can be simulated by a random access machine (a RAM) in $O(n \cdot \log n)$ steps. As each operation involves only $\log n$ many bits, we obtain the following result.

Theorem 18 *The membership problem for a nrDFAwtl is decidable in time* $\mathrm{O}(n \cdot (\log n)^2)$.

The construction shows that each operation of a non-returning NFAwtl can be simulated nondeterministically by a RAM in $\log n$ many steps. Hence, we obtain the following obvious corollary.

Corollary 19 *The membership problem for an nrNFAwtl is in* $\mathsf{NTIME}(n \cdot (\log n)^2)$.

In fact, by using an extension of the technique presented by Nagy and Kovács in [10], it can be shown that the membership problem for a nrDFAwtl is even decidable in time $O(n \cdot \log n)$ if the underlying alphabet is only of cardinality two. However, it remains open whether a corresponding result can also be obtained for the case of larger alphabets.

By Proposition 12, a unary language is accepted by an nrNFAwtl if and only if it is a regular language. This implies immediately that the emptiness problem (and the finiteness problem) is decidable for nrNFAwtls that accept unary languages. Concerning non-unary languages, the situation is more complicated.

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. In this case, each accepting computation consists of a sequence of sweeps. Now 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 20 *Let* $A = (Q, \Sigma, \lhd, \tau, q_0, \delta)$ *be the nrDFAwtl that is defined 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:*

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

This nrDFAwtl is depicted by the diagram in Figure 3.

$$\begin{tikzcd} {} \arrow[r] & q_0 \arrow[r, "(b^*,a)"] & q_1 \arrow[r, "(c^*,\lhd)"] & q_2 \arrow[r, "(c^*,b)"] & q_3 \arrow[r, "\lhd"] & q_4 \arrow[r, "(a^*,c)"] & q_5 \arrow[r, "\lhd"] & \mathsf{Accept} \end{tikzcd}$$

Figure 3: The nrDFAwtl A from Example 20

From this diagram, we can immediately extract three sweeps (or rather cycles):

$$\begin{array}{llllll}
(1) & q_0 b^* a c^* \cdot \lhd & \vdash_A & b^* q_1 c^* \cdot \lhd & \vdash_A & q_2 b^* c^* \cdot \lhd, \\
(2) & q_2 c^* b \cdot \lhd & \vdash_A & c^* q_3 \lhd & \vdash_A & q_4 c^* \cdot \lhd, \\
(3) & q_4 a^* c \cdot \lhd & \vdash_A & a^* q_5 \lhd & \vdash_A & \mathsf{Accept}.
\end{array}$$

However, these three sweeps (or cycles) cannot be combined into an accepting computation of A. The third sweep requires that there is an occurrence of the letter c which may only be preceded by occurrences of the letter a. The second sweep requires that there is an occurrence of the letter b, which may only be preceded by occurrences of the letter c. Finally, the first sweep 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 an occurrence 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.

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. The non-returning types of automata are indeed more expressive than the original types. In fact, we presented a complete classification of the resulting language classes in relation to the Chomsky hierarchy. Also we derived some closure and non-closure properties for these language classes and proved that the membership problem for a non-returning DFAwtl is of time complexity $O(n \cdot (\log n)^2)$.

However, many questions concerning the nrNFAwtl and the nrDFAwtl are still open. Here we stress only four of them.

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 $\mathscr{L}(\mathsf{nrNFAwtl})$ closed under intersection with regular sets? If it is, then we see from Example 13 that $\mathscr{L}(\mathsf{nrNFAwtl})$ contains languages that are not semi-linear. However, we conjecture that $\mathscr{L}(\mathsf{nrNFAwtl})$ is not closed under this operation, as we expect that the language $L_{\exp3}$ is not accepted by any nrNFAwtl.
3. Can the upper bound of $O(n \cdot (\log n)^2)$ for the time complexity of the membership problem for a nrDFAwtl be improved to $O(n \cdot \log n)$ also for alphabets of cardinality larger than two?
4. Is emptiness decidable for nrNFAwtls or for nrDFAwtls?

Currently, pumping techniques as used in the proofs of Proposition 14 and Theorem 15 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.

References

[1] J.-M. Autebert, J. Berstel & L. Boasson (1997): *Context-free languages and pushdown automata*. In G. Rozenberg & A. Salomaa, editors: *Handbook of Formal Languages*, 1, Springer, Berlin, Heidelberg, pp. 111–174, doi:10.1007/978-3-642-59136-5_3.

[2] G. Buntrock & F. Otto (1998): *Growing context-sensitive languages and Church-Rosser languages*. *Informormation and Computation* 141, pp. 1–36, doi:10.1006/inco.1997.2681.

[3] T.H. Cormen, C.E. Leiserson, R.L. Rivest & C. Stein (2022): *Introduction to Algorithms*, 4th edition. MIT Press.

[4] H. Fernau, M. Paramasivan & M.L. Schmid (2012): *Jumping finite automata: Characterizations and complexity*. In F. Drewes, editor: *CIAA 2012, Proc., Lecture Notes in Computer Science* 9223, Springer, Heidelberg, pp. 89–101, doi:10.1007/978-3-319-22360-5_8.

[5] P. Jančar, F. Mráz, M. Plátek & J. Vogel (1995): *Restarting automata*. In H. Reichel, editor: *FCT'95, Proc., Lecture Notes in Computer Science* 965, Springer, Berlin, pp. 283–292, doi:10.1007/3-540-60249-6_60.

[6] R. Loukanova (2007): *Linear context free languages*. In C.B. Jones, Z. Liu & J. Woodcock, editors: *ICTAC 2007, Proc.*, *Lecture Notes in Computer Science* 4711, Springer, Heidelberg, pp. 351–365, doi:10.1007/978-3-540-75292-9_24.

[7] A. Meduna & P. Zemek (2012): *Jumping finite automata*. *International Journal of Foundations of Computer Science* 23, pp. 1555–1578, doi:10.1142/S0129054112500244.

[8] F. Mráz (2001): *Lookahead hierarchies of restarting automata*. *Journal of Automata, Languages and Combinatorics* 6, pp. 493–506, doi:10.25596/jalc-2001-493.

[9] B. Nagy (2008): *On $5' \rightarrow 3'$ sensing Watson-Crick automata*. In M. Garzon & H. Yan, editors: *DNA Computing, 13th Intern. Meeting, Revised Selected Papers*, *Lecture Notes in Computer Science* 4848, Springer, Heidelberg, pp. 256–262, doi:10.1007/978-3-540-77962-9_27.

[10] B. Nagy & L. Kovács (2014): *Finite Automata with Translucent Letters Applied in Natural and Formal Language Theory*. In N.T. Nguyen, R. Kowalczyk, A. Fred & F. Joaquim, editors: *Transactions on Computational Collective Intelligence XVII*, *Lecture Notes in Computer Science* 8790, Springer, Heidelberg, pp. 107–127, doi:10.1007/978-3-662-44994-3_6.

[11] B. Nagy & F. Otto (2010): *CD-systems of stateless deterministic R(1)-automata accept all rational trace languages*. In A.H. Dediu, H. Fernau & C. Martin-Vide, editors: *LATA 2010, Proc.*, *Lecture Notes in Computer Science* 6031, Springer, Berlin, pp. 463–474, doi:10.1007/978-3-642-13089-2_39.

[12] B. Nagy & F. Otto (2011): *Finite-state acceptors with translucent letters*. In G. Bel-Enguix, V. Dahl & A.O. De La Puente, editors: *BILC 2011: AI Methods for Interdisciplinary Research in Language and Biology, Proc.*, SciTePress, Portugal, pp. 3–13.

[13] B. Nagy & F. Otto (2011): *Globally deterministic CD-systems of stateless R(1)-automata*. In A.H. Dediu, S. Inenaga & C. Martín-Vide, editors: *Language and Automata Theory and Applications, LATA 2011, Proc.*, *Lecture Notes in Computer Science* 6638, Springer, Berlin, pp. 390–401, doi:10.1007/978-3-642-21254-3_31.

[14] B. Nagy & F. Otto (2012): *On CD-systems of stateless deterministic R-automata with window size one*. *Journal of Computer and System Sciences* 78, pp. 780–806, doi:10.1016/j.jcss.2011.12.009.

[15] B. Nagy & F. Otto (2013): *Globally deterministic CD-systems of stateless R-automata with window size 1*. *International Journal of Computer Mathematics* 90, pp. 1254–1277, doi:10.1080/00207160.2012.688820.

[16] F. Otto (2006): *Restarting automata*. In Z. Ésik, C. Martín-Vide & V. Mitrana, editors: *Recent Advances in Formal Languages and Applications*, *Studies in Computational Intelligence* 25, Springer, Heidelberg, pp. 269–303.