RAIRO-Theor. Inf. Appl. 54 (2020) 3
https://doi.org/10.1051/ita/2020002

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

LINEAR AUTOMATA WITH TRANSLUCENT LETTERS AND LINEAR CONTEXT-FREE TRACE LANGUAGES*

Benedek Nagy[1] and Friedrich Otto[2,**]

Abstract. Linear automata with translucent letters are studied. These are finite-state acceptors that have two heads that read the input from opposite sides and for which a set of translucent letters is associated with each state. Thus, head 1, which proceeds from left to right, does not necessarily read the first letter of the current tape content, but it skips a prefix that consists of translucent letters only and reads the first letter after that prefix. Analogously, head 2, which proceeds from right to left, does not necessarily read the last letter, but it skips a suffix that consists of translucent letters only and reads the last letter before that. After such a read operation, the head always returns to its corresponding end of the tape. These linear automata with translucent letters are a generalization of the finite-state acceptors with translucent letters that were studied by the authors in B. Nagy and F. Otto [Finite-state acceptors with translucent letters. In *BILC 2011, Proc.*, edited by G. Bel-Enguix, V. Dahl, and A.O. De La Pente, SciTePress, Portugal (2011) 3-13.] It is shown that these linear automata are strictly more expressive than the model with a single head, but that they still only accept languages that have a semi-linear Parikh image. On the other hand, we obtain a characterization for the class of linear context-free trace languages in terms of a specific class of linear automata with translucent letters.

Mathematics Subject Classification. 68Q45.

Received April 30, 2019. Accepted February 18, 2020.

1. Introduction

The finite-state acceptor is one of the most fundamental computing devices for accepting languages. Its deterministic version (DFA) and its nondeterministic version (NFA) both accept exactly the regular languages, and they have found applications in many areas like compiler construction, text editors, hardware design, etc.

A finite-state acceptor reads its input strictly sequentially from left to right, letter by letter. However, in the literature one finds many extensions of this model that process their inputs in different ways. Here we just mention some of them:

*Some of the results of this paper have been announced at SOFSEM 2019 in Nový Smokovec, Slovakia, January 2019. An extended abstract appeared in the proceedings of that conference [28].

Keywords and phrases: Linear automaton, translucent letter, linear context-free trace language.

[1] Department of Mathematics, Faculty of Arts and Sciences, Eastern Mediterranean University, Famagusta, North Cyprus, Via Mersin-10, Turkey.
[2] Fachbereich Elektrotechnik/Informatik, Universität Kassel, 34109 Kassel, Germany.

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

- The *multi-head finite-state acceptor*, which has a finite number of heads that all read the input from left to right, can easily compare different parts of the input to each other without the necessity of storing these parts in its finite-state control [33]. Accordingly, this type of automaton accepts some rather complex languages like the copy language that is not even growing context-sensitive (see, *e.g.*, [5]).
- The *Watson-Crick automaton* (see, *e.g.*, [6, 10, 32]), which has two heads that read the input from left to right, but which works on double stranded words where letters on corresponding positions are connected by a complementarity relation. Actually, the Watson Crick automaton is equivalent to the two-head finite-state acceptor.
- The *nondeterministic linear automaton* [15], which has two heads that start from the two ends of an input word, one reading the word from left to right and the other reading it from right to left, halting when the two heads meet. These automata characterize the class of linear context-free languages. They correspond to the $5' \to 3'$ *sensing Watson-Crick automaton* studied in [14, 17, 19, 21, 30] and to a class of 2-head finite automata for linear languages [18].
- The finite-state acceptor with translucent letters [20, 25], which has a single head starting at the left end of an input word, but depending on the actual state, it skips across a prefix of letters that are translucent, in this way reading (and deleting) a letter from the input. This type of automaton is equivalent to the cooperating distributed systems of stateless deterministic R(1)-automata that were introduced and studied in [22]. They only accept languages with a semi-linear Parikh image, but they do accept all rational trace languages [26]. In fact, the rational trace languages can be characterized by a certain restricted class of these automata.
- The *jumping automaton* [16], which has a single head starting at the left end of the input, but that jumps in each step to an arbitrary position reading (and deleting) the letter at that position. A detailed study of the expressive power of this model can be found in [9]. We note that there is also a model which combines the features of jumping automata and $5' \to 3'$ sensing Watson-Crick automata [13].

In this paper we propose a new type of two-head finite-state acceptor, the *nondeterministic linear automaton with translucent letters*[1] (NLAwtl, for short). It is obtained by combining the concept of the nondeterministic linear automaton with the idea of translucent letters from the finite-state acceptor with translucent letters. Such a device is given an input word surrounded by sentinels, and it has two heads that start at the two ends of a given input word, being positioned on the sentinels, one scanning the input from left to right, the other scanning it from right to left. However, depending on the actual state, certain letters are translucent, that is, the left head, that is, the one scanning the input from left to right, skips across a prefix of translucent letters and reads (and deletes) the first letter that is not translucent for the current state, and analogously, the right head, that is, the head that scans the input from right to left, skips across a suffix of translucent letters and reads (and deletes) the first letter from the right that is not translucent for the current state. If no such letter is found, then the automaton halts, accepting if the current state is final.

As the NLAwtl extends the nondeterministic linear automaton, it accepts all linear context-free languages. Actually, the NLAwtl even accepts some languages that are not context-free, but we will see that each language L accepted by an NLAwtl contains a linear context-free sublanguage that is letter-equivalent to L. This implies in particular that all these languages have semi-linear Parikh images. Further, we will see that all linear context-free trace languages are accepted by NLAwtls, and we can even characterize this class of trace languages by a restricted type of NLAwtls. In addition, we will see that the DLAwtl, the deterministic variant of the NLAwtl, is less expressive than the NLAwtl, and we establish some closure and some non-closure properties for the classes of languages accepted by NLAwtls and by DLAwtls. Also we consider a number of decision problems for NLAwtls.

[1] In [28] these automata are called *two-head finite-state acceptors with translucent letters*, abbreviated as 2hNFAwtl, but as our automata are derived from the nondeterministic linear automata, the new name seems to fit better and it is in addition simpler.

Finally, we consider the class of NLAwtls $\mathcal{A}(A)$ that is obtained by taking a fixed nondeterministic linear automaton A and by then adding all admissible translucency relations and study the corresponding class of languages $\mathcal{L}(\mathcal{A}(A))$. It is easily seen that $\mathcal{A}(A)$ forms a lattice with respect to the inclusion relation on translucency relations, but what can be said about the corresponding class of languages?

This paper is structured as follows. In Section 2, we recall basic notions and notation of formal language theory. In Section 3, we introduce the nondeterministic linear automaton with translucent letters and its deterministic variant, we present an example illustrating the expressive power of NLAwtls, and we establish a normal form for NLAwtls. In Section 4, we derive the aforementioned results on the expressive power of the NLAwtl. Then we study closure and decidability results in Section 5 and introduce the classes $\mathcal{A}(A)$ of NLAwtls in Section 6.

2. Preliminaries

For a finite alphabet Σ, we use Σ^+ to denote the set of non-empty words over Σ and Σ^* to denote the set of all words over Σ including the empty word ε. For a word $w \in \Sigma^*$, $|w|$ denotes the length of w, and $|w|_a$ is the a-length of w, that is, the number of occurrences of the letter a in w. For a subset Γ of Σ, $|w|_\Gamma = \sum_{a\in\Gamma} |w|_a$, and $\pi_\Gamma : \Sigma^* \to \Gamma^*$ is the projection from Σ^* onto Γ^*, that is, π_Γ is the morphism defined by $\pi_\Gamma(a) = a$ for all $a \in \Gamma$ and $\pi_\Gamma(b) = \varepsilon$ for all $b \in \Sigma \setminus \Gamma$. Further, for a set S, we use the notation 2^S for the power set of S. Finally, for any automaton A, $L(A)$ will denote the language that consists of all words that are accepted by A, and for any type of automaton $\mathcal{A}$, $\mathcal{L}(\mathcal{A})$ is the class of languages that are accepted by automata of type $\mathcal{A}$.

Here we assume that the reader is familiar with the basics of formal language and automata theory for which we refer to the textbooks [12, 35] and to the Handbook [34]. By REG, LIN, CFL, and GCSL we denote the classes of regular, linear context-free, context-free and growing context-sensitive languages, respectively. For the latter we refer to [7].

A *nondeterministic linear automaton* (NLA) [15] is described by a 7-tuple $A = (Q, \Sigma, \delta, q_0, L, R, F)$, where Q is a finite set of internal states, Σ is a finite input alphabet, $q_0 \in Q$ is the initial state, $L, R \notin Q \cup \Sigma$ are special symbols that will be used to mark the positions of the left and right head, respectively, $F \subseteq Q$ is the set of final (or accepting) states, and $\delta : Q \times \Sigma \to 2^{Q\times\{L,R\}}$ is a transition relation.

An NLA A works as follows. On an input word $w \in \Sigma^*$, it starts in its initial state q_0 with its first (or left) head on the first letter of w and its second (or right) head on the last letter of w. This configuration is encoded as $\langle q_0, LwR\rangle$. Now, depending on the allowed transitions, it reads the first or the last letter of w, say a or b, moves the corresponding head to the next letter, and changes its state. Formally, the configuration $\langle q, LavbR\rangle$ can be transformed into the configuration $\langle p, LvbR\rangle$ if $(p, L) \in \delta(q, a)$, and it can be transformed into the configuration $\langle p', LavR\rangle$ if $(p', R) \in \delta(q, b)$, where $p, p', q \in Q$ and $a, b \in \Sigma$. If no transition can be applied, then A gets stuck, that is, it halts without accepting if $w = avb$ is nonempty. Otherwise, it continues reading (and deleting) letters until w has been consumed completely. Thus, from a configuration of the form $\langle q, LaR\rangle$ we reach the halting configuration $\langle p, RL\rangle$ if $(p, L) \in \delta(q, a)$ or $(p, R) \in \delta(q, a)$. The configuration $\langle p, RL\rangle$ is accepting if p is a final state. We say that A accepts a word w if A has a computation that is in a final state $q_f \in F$ after reading w completely. In addition, we say that A accepts the empty word ε if $q_0 \in F$. By $L(A)$ we denote the set of all words $w \in \Sigma^*$ for which A has an accepting computation in the sense described above.

It is known that the class $\mathcal{L}(\mathsf{NLA})$ of languages $L(A)$ that are accepted by NLAs coincides with the class LIN of linear context-free languages [15].

For later use we modify the description of the NLA as follows. First of all, instead of a single initial state q_0, we admit a finite set $I \subseteq Q$ of initial states. It is easily seen that an NLA $A = (Q, \Sigma, \delta, I, L, R, F)$ can be turned into an equivalent NLA $B = (Q \cup \{q_0\}, \Sigma, \delta_B, q_0, L, R, F)$ by defining δ_B as follows, where $a \in \Sigma$:

$$\delta_B(q_0, a) = \bigcup_{q\in I} \delta(q, a) \text{ and } \delta_B(q, a) = \delta(q, a) \text{ for all } q \in Q.$$

Next, instead of leaving the choice of whether to use the left or the right head to the moment at which a transition is to be applied, we shift this choice to the previous transition. This is done by partitioning the set of states Q of an NLA A into two subsets, the set Q_L of *left states* and the set Q_R of *right states*, and by defining δ simply as a transition relation $\delta : Q \times \Sigma \to 2^Q$. Let $\langle q, LavbR\rangle$ be a configuration of A. If q is a left state, then a state $p \in \delta(q,a)$ must be chosen and the configuration $\langle p, LvbR\rangle$ is reached, and if q is a right state, then a state $p' \in \delta(q,b)$ must be chosen and the configuration $\langle p', LavR\rangle$ is reached. By splitting each state q of an NLA $A = (Q, \Sigma, \delta, I, L, R, F)$ into a left state q_L and a right state q_R and by defining $\delta_B(q_L, a) = \{\, p_L, p_R \mid (p, L) \in \delta(q,a) \,\}$ and $\delta_B(q_R, a) = \{\, p_L, p_R \mid (p, R) \in \delta(q,a) \,\}$, we obtain an NLA $B = (Q_L \cup Q_R, \Sigma, \delta_B, I_L \cup I_R, L, R, F_L \cup F_R)$ that accepts the same language as A. Here $Q_L = \{\, q_L \mid q \in Q \,\}$, $Q_R = \{\, q_R \mid q \in Q \,\}$, and analogously for the subsets I and F of Q. As each state of B is either a left or a right state, we actually do not need the special marks L and R anymore. Accordingly, we will use the following definition for the NLA in this paper.

Definition 2.1.

(a) A *nondeterministic linear automaton* (*NLA*) is described by a 5-tuple $A = (Q, \Sigma, \delta, I, F)$, where Q is a finite set of internal states that is partitioned into two disjoint subsets $Q = Q_L \cup Q_R$ of *left states* and *right states*, Σ is a finite input alphabet, $I \subseteq Q$ is the set of initial states, $F \subseteq Q$ is the set of final (or accepting) states, and $\delta : Q \times \Sigma \to 2^Q$ is a transition relation.
For an input word $w \in \Sigma^*$, a corresponding initial configuration is simply written as $q_0 w$, where $q_0 \in I$. The single-step transition relation $\vdash_A$ that is induced by A on its set of configurations is defined as follows, where $a, b \in \Sigma$:

$$qavb \vdash_A \begin{cases} pvb, & \text{if } q \in Q_L \text{ and } p \in \delta(q,a), \\ p'av, & \text{if } q \in Q_R \text{ and } p' \in \delta(q,b), \\ \text{undefined}, & \text{otherwise.} \end{cases}$$

By $\vdash_A^*$ we denote the computation relation of A, which is the reflexive and transitive closure of $\vdash_A$.

(b) An NLA $A = (Q, \Sigma, \delta, I, F)$ is said to be in *normal form* if

$$|\mu(q)| \leq 1 \text{ for each state } q \in Q,$$

where $\mu(q) = \{\, a \in \Sigma \mid \delta(q,a) \neq \emptyset \,\}$, that is, for each state $q \in Q$, there exists at most one letter $a \in \Sigma$ such that $\delta(q,a)$ is defined.

(c) An NLA $A = (Q, \Sigma, \delta, I, F)$ is a *deterministic linear automaton* (*DLA*) if it has only a single initial state, that is, $|I| = 1$, and if $|\delta(q,a)| \leq 1$ for each state $q \in Q$ and each letter $a \in \Sigma$.

For each NLA B, there exists an NLA A in normal form such that $L(A) = L(B)$. In fact, if $\mu(q) = \{a_1, a_2, \ldots, a_m\}$, then one can simply replace the state q by the new states $q_{i_1}, q_{i_2}, \ldots, q_{i_m}$ such that $\mu(q_{i_j}) = \{a_j\}$ for all $j = 1, 2, \ldots, m$, and in the transition relation δ, if $q \in \delta(p,a)$, then q is replaced by all states $q_{i_1}, q_{i_2}, \ldots, q_{i_m}$, and analogously in I and in F. So we can restrict our attention to NLAs that are in normal form.

It is easily seen that our definition of the DLA is equivalent to the definition of the DLA given in [15] and, hence, it defines the language class 2detLIN [19, 29, 31]. The reason is the following observation. If $\delta(q,a) = (p, L)$ and $\delta(q,b) = (p', R)$ for some states q, p, p' and some letters $a, b \in \Sigma$, then the DLA $A = (Q, \Sigma, \delta, q_0, L, R, F)$ would have two applicable transitions in a configuration of the form $\langle q, LavbR\rangle$. Hence, for each state, it either can only use the left head (that is, this state can be interpreted as a left state) or the right head (that is, this state can be interpreted as a right state). In [15, 17] it has been observed that the language class $\mathcal{L}(\mathsf{DLA})$ is incomparable under inclusion to the class DCFL of deterministic context-free languages, as, *e.g.*, the language $\{\, a^n b^n c, a^n b^{2n} d \mid n \geq 1 \,\}$, which is known to not be deterministic context-free, is accepted by a DLA, while it can be shown that the deterministic context-free language $\{\, a^n c b^n, a^n b^{2n} \mid n \geq 1 \,\}$ is not accepted by any DLA.

In fact, the latter language is accepted by a deterministic one-turn pushdown automaton, that is, it is even a deterministic linear language (see, *e.g.*, [2]). Thus, the language class $\mathcal{L}(\mathsf{DLA})$ is in fact incomparable under inclusion to the class of deterministic linear languages.

Next we recall the definition of the finite-state acceptor with translucent letters. A *finite-state acceptor with translucent letters* (NFAwtl) is defined by a 7-tuple $A = (Q, \Sigma, \lhd, \tau, I, F, \delta)$, where Q is a finite set of internal states, Σ is a finite input alphabet, $\lhd \notin \Sigma$ is a special symbol that is used as an *endmarker*, $\tau : Q \to 2^{\Sigma}$ is a *translucency mapping*, $I \subseteq Q$ is a set of initial states, $F \subseteq Q$ is a set of final states, and $\delta : Q \times \Sigma \to 2^{Q}$ is a *transition relation*. 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 cannot see them.

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 $w\lhd$ on its input tape. Assume that $w = a_1 a_2 \cdots a_n$ for some $n \geq 1$ and $a_1, a_2, \ldots, a_n \in \Sigma$. Then A looks for the first occurrence from the left of a letter that is visible, that is, the leftmost letter of w that is not translucent for state q_0. Thus, if $w = uav$ such that $u \in (\tau(q_0))^*$ and $a \notin \tau(q_0)$, then A nondeterministically chooses a state $q_1 \in \delta(q_0, a)$, erases the letter a from the tape thus producing the tape contents $uv\lhd$, and its internal state is set to q_1. In case $\delta(q_0, a) = \emptyset$, A halts without accepting. Finally, if $w \in (\tau(q_0))^*$, then A reaches the $\lhd$-symbol and the computation halts. In this case A accepts if q_0 is a final state; otherwise, it does not accept. Thus, A executes the following computation relation on its set $Q \cdot \Sigma^* \lhd$ of configurations:

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

Observe that this definition also applies to configurations of the form $q\lhd$, that is, when the tape just contains the endmarker. In this situation we see that $q\lhd \vdash_A \mathsf{Accept}$ 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 $q_0 \in I$ and a computation $q_0 w\lhd \vdash_A^* \mathsf{Accept}$, 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.

Obviously, NFAwtls accept all regular languages, as the classical nondeterministic finite-state acceptor, the NFA, is obtained from the NFAwtl by removing the endmarker $\lhd$, by ignoring the translucency relation τ, and by adding a new initial state from which there are ε-transitions to all states from the set I.

3. Linear automata with translucent letters

Now we are ready to introduce our new model based on linear automata and NFAwtls.

Definition 3.1. A *nondeterministic linear automaton with translucent letters* (*NLAwtl*) consists of a finite-state control, a single flexible tape with endmarkers, and two heads that are positioned on these endmarkers. It is defined by an 8-tuple $A = (Q, \Sigma, \rhd, \lhd, \tau, \delta, I, F)$, where

- Q is a finite set of states that is partitioned into two subsets $Q = Q_L \cup Q_R$ of *left states* and *right states*,
- Σ is a finite input alphabet,
- $\rhd, \lhd \notin \Sigma$ are special symbols that are used as *endmarkers*, that is, the tape contents is being written between $\rhd$ and $\lhd$,
- $\tau : Q \to 2^{\Sigma}$ is a *translucency mapping*,
- $I \subseteq Q$ is a set of initial states,
- $F \subseteq Q$ is a set of final states, and
- $\delta : Q \times \Sigma \to 2^{Q}$ is a transition relation.

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 cannot see these letters.

A is called a *deterministic linear automaton with translucent letters*, abbreviated as *DLAwtl*, if $|I| = 1$ and if $|\delta(q,a)| \leq 1$ for all $q \in Q$ and all $a \in \Sigma$.

An NLAwtl $A = (Q, \Sigma, \rhd, \lhd, \tau, \delta, I, F)$ works as follows. For an input word $w \in \Sigma^*$, an initial configuration consists of the automaton being in an initial state q_0 chosen nondeterministically from the set I with the word $\rhd w \lhd$ on its tape. Assume that $w = a_1 a_2 \cdots a_n$ for some $n \geq 1$ and $a_1, a_2, \ldots, a_n \in \Sigma$. If $q_0 \in Q_L$, that is, q_0 is a left state, then A looks for the first occurrence from the left of a letter that is not translucent for state q_0, and if $q_0 \in Q_R$, that is, q_0 is a right state, then A looks for the first occurrence from the right (that is, the last occurrence) of a letter that is not translucent for state q_0:

- **Left-reading:** If $q_0 \in Q_L$, then A proceeds as follows. If $w = uaz$ for some $u \in (\tau(q_0))^*$ and $a \in \Sigma \smallsetminus \tau(q_0)$, then A nondeterministically chooses a state $q_1 \in \delta(q_0, a)$, erases the letter a from the tape, in this way producing the tape contents $\rhd uz \lhd$, returns its left head to the left delimiter $\rhd$, and sets its finite-state control to q_1. If $\delta(q_0, a) = \emptyset$, then A gets stuck, that is, it halts without accepting. If $w \in (\tau(q_0))^*$, then A halts, and it accepts if q_0 is a final state, that is, if $q_0 \in F$.

- **Right-reading:** If $q_0 \in Q_R$, then A proceeds as follows. If $w = vby$ for some $y \in (\tau(q_0))^*$ and $b \in \Sigma \smallsetminus \tau(q_0)$, then A nondeterministically chooses a state $q_1 \in \delta(q_0, b)$, erases the letter b from the tape, in this way producing the tape contents $\rhd vy \lhd$, returns its right head to the right delimiter $\lhd$, and sets its finite-state control to q_1. If $\delta(q_0, b) = \emptyset$, then A gets stuck, that is, it halts without accepting. If $w \in (\tau(q_0))^*$, then A halts, and it accepts if q_0 is a final state, that is, if $q_0 \in F$.

Thus, A executes the following computation relation on its set $Q \cdot \rhd \Sigma^* \lhd$ of configurations:

$$q \rhd w \lhd \vdash_A \begin{cases} q' \rhd uz \lhd, & \text{if } q \in Q_L \text{ and } w = uaz \text{ for } u \in (\tau(q))^*, a \in \Sigma \smallsetminus \tau(q), \\ & \text{and } q' \in \delta(q,a), \\ \mathsf{Reject}, & \text{if } q \in Q_L \text{ and } w = uaz \text{ for } u \in (\tau(q))^*, a \in \Sigma \smallsetminus \tau(q), \\ & \text{and } \delta(q,a) = \emptyset, \\ q' \rhd vy \lhd, & \text{if } q \in Q_R \text{ and } w = vby \text{ for } y \in (\tau(q))^*, b \in \Sigma \smallsetminus \tau(q), \\ & \text{and } q' \in \delta(q,b), \\ \mathsf{Reject}, & \text{if } q \in Q_R \text{ and } w = vby \text{ for } y \in (\tau(q))^*, b \in \Sigma \smallsetminus \tau(q), \\ & \text{and } \delta(q,b) = \emptyset, \\ \mathsf{Accept}, & \text{if } w \in (\tau(q))^* \text{ and } q \in F, \\ \mathsf{Reject}, & \text{if } w \in (\tau(q))^* \text{ and } q \notin F. \end{cases}$$

Observe that this definition also applies to configurations of the form $q \rhd \lhd$, that is, $q \rhd \lhd \vdash_A \mathsf{Accept}$ 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 $q_0 \in I$ and a computation $q_0 \rhd w \lhd \vdash_A^* \mathsf{Accept}$, where $\vdash_A^*$ denotes the reflexive 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, $\mathcal{L}(\mathsf{NLAwtl})$ denotes the class of languages that are accepted by NLAwtls, and $\mathcal{L}(\mathsf{DLAwtl})$ denotes the class of languages that are accepted by DLAwtls.

If A is an NLAwtl such that $\tau(q) = \emptyset$ for all states q of A, then A is essentially just an NLA. Conversely, if B is an NLA, then B can be seen as an NLAwtl A with empty transparency sets and $L(A) = L(B)$. Hence, the NLAwtl is an extension of the NLA, which shows immediately that $\mathsf{LIN} \subseteq \mathcal{L}(\mathsf{NLAwtl})$.

On the other hand, the NFAwtls obviously correspond to those NLAwtls for which all states are left states. Thus, the NLAwtl is also an extension of the NFAwtl. It is known that the linear language $\{ a^n b^n \mid n \geq 0 \}$ is not accepted by any NFAwtl [25], while $\mathcal{L}(\mathsf{NFAwtl})$ does even contain some non-context-free languages, for

example, the language $\{ w \in \{a,b,c\}^* \mid |w|_a = |w|_b = |w|_c \}$. However, the NLAwtl is more expressive than the NFAwtl, which is shown by the following interesting example.

Example 3.2. Let L be the language

$$L \quad = \quad \{ w_1 \# u \# w_2 \quad | \quad w_1, w_2 \in \{a,b\}^*, |w_1|_a = |w_2|_a, |w_1|_b = |w_2|_b, \text{ and } u \in \{c,d\}^* \text{ is a palindrome} \}.$$

Obviously, L is not context-free, as $L \cap a^* \cdot b^* \cdot \# \cdot \{c,d\}^* \cdot \# \cdot a^* \cdot b^* = \{ a^m b^n \# u \# a^m b^n \mid m,n \geq 0 \text{ and } u \in \{c,d\}^* \text{ is a palindrome} \}$. Further, L does not contain any regular subset that is letter-equivalent to L itself, and hence, L is not accepted by any NFAwtl [25]. However, L is accepted by the DLAwtl $A = (Q, \Sigma, \rhd, \lhd, \tau, \delta, I, F)$ that is defined as follows:

- $Q = Q_L \cup Q_R$, where $Q_L = \{q_a, q_c\}$ and $Q_R = \{p_a, p_b, p_\#, p_c, p_d\}$,
- $\Sigma = \{a, b, c, d, \#\}$,
- $I = \{q_a\}$ and $F = \{q_c, p_c, p_d\}$,
- τ is defined by the following table:

q_a	q_c	p_a	p_b	$p_\#$	p_c	p_d
$\emptyset$	$\emptyset$	$\{b\}$	$\{a\}$	$\emptyset$	$\emptyset$	$\emptyset$

,

- and the transition relation is given through the following table:

	q_a	q_c	p_a	p_b	$p_\#$	p_c	p_d
a	p_a	–	q_a	–	–	–	–
b	p_b	–	–	q_a	–	–	–
$\#$	$p_\#$	–	–	–	q_c	–	–
c	–	p_c	–	–	–	q_c	–
d	–	p_d	–	–	–	–	q_c

Let $w = x\#y\#z$ be given as input, where $x, z \in \{a,b\}^*$ and $y \in \{c,d\}^*$. Then A starts with the initial configuration $q_a \rhd x\#y\#z\lhd$. If $x \neq \varepsilon$, then A reads the first letter of x, say a, deletes this letter, and enters state p_a. Now A will get stuck if $|z|_a = 0$, otherwise it will delete the rightmost a occurring in z and then again enter its initial state q_a. Analogously, if the first letter of x is a b, then A enters state p_b. Now A will get stuck if $|z|_b = 0$, otherwise it will delete the rightmost b occurring in z and then again enter its initial state q_a. Thus, by going through states q_a and then p_a or p_b, one occurrence of the letter a or b has been removed from x and from z. If $|x|_a > |z|_a$ or $|x|_b > |z|_b$, then A will eventually detect this and it will then get stuck in state p_a or p_b. Otherwise, after erasing the prefix x, A reads the letter $\#$ and enters state $p_\#$. Using this state it checks that the remaining tape contents ends with the symbol $\#$, which it deletes in the affirmative entering state q_c. Thus, if $|z|_a > |x|_a$ or $|z|_b > |x|_b$, then A gets stuck in state $p_\#$. If, however, $|z|_a = |x|_a$ and $|z|_b = |x|_b$, then A enters state q_c, and using this state and the states p_c and p_d, it checks whether y is a palindrome over $\{c,d\}$. In the negative, it gets stuck, while in the affirmative, it will read and delete y completely and accept. It follows that $L(A) = L$.

Thus, we have derived the following proper inclusions.

Proposition 3.3. $\mathcal{L}(\mathsf{NLA}) = \mathsf{LIN} \subsetneq \mathcal{L}(\mathsf{NLAwtl})$ *and* $\mathcal{L}(\mathsf{NFAwtl}) \subsetneq \mathcal{L}(\mathsf{NLAwtl})$.

An NLAwtl $A = (Q, \Sigma, \rhd, \lhd, \tau, \delta, I, F)$ can be described more transparently by a graph, similar to the graph representation of an NFA. A state $q \in Q$ is represented by a node, which is labelled with $q^{\rightarrow}$ if $q \in Q_L$, as such a state refers to the left head that reads the tape from left to right, and which is labelled with $q^{\leftarrow}$ if $q \in Q_R$,

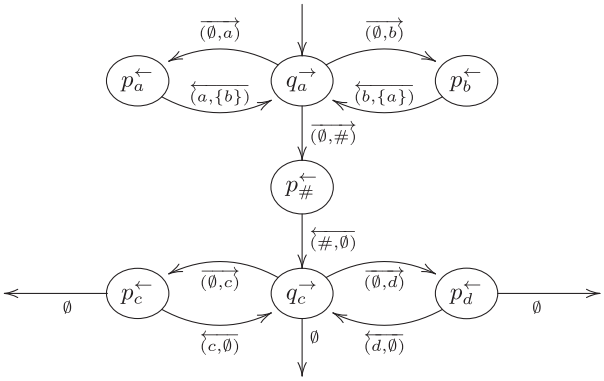


FIGURE 1. The graphical representation of the DLAwtl A of Example 3.2.

as such a state refers to the right head that reads the tape from right to left. In addition, the node of an initial state q is marked by a special incoming edge without a label, and the node of a final state q is marked by a special outgoing edge with label $\tau(q)$. Further, if $q \in Q$ and $p \in \delta(q,a)$ for some letter $a \in \Sigma$, then there is a directed edge from the node with label q to the node with label p. This edge is labelled with $\overrightarrow{(\tau(q),a)}$ if $q \in Q_L$, and it is labelled with $\overleftarrow{(a,\tau(q))}$ if $q \in Q_R$. For example, the DLAwtl A from Example 3.2 can be depicted by the diagram in Figure 1.

Definition 3.4. Let $A = (Q, \Sigma, \rhd, \lhd, \tau, \delta, I, F)$ be an NLAwtl. For each state $q \in Q$ we define $\mu(q) = \{\, a \in \Sigma \mid \delta(q,a) \neq \emptyset \,\}$, that is, $\mu(q)$ is the set of letters which A can read in state q. Observe that we can assume without loss of generality that $\mu(q) \cap \tau(q) = \emptyset$ for all states $q \in Q$. Now the NLAwtl A is said to be in *normal form* if

1. $|\mu(q)| \leq 1$ for each state $q \in Q$, that is, for each state $q \in Q$, there exists at most one letter $a \in \Sigma$ such that $\delta(q,a)$ is defined,
2. it always accepts with empty tape, that is, each word from $L(A)$ is read (and deleted) completely before A accepts.

By splitting the state q_a into three different states q_a, q_b, and $q_\#$ and by splitting the state q_c into two different states q_c and q_d such that, in state q_x, only the letter x can be read for all $x \in \{a,b,\#,c,d\}$, the DLAwtl A from Example 3.2 can be transformed into an NLAwtl that is in normal form. Concerning NLAwtls, we now derive the following result in analogy to the situation for NLAs.

Proposition 3.5. *From a given NLAwtl $A = (Q, \Sigma, \rhd, \lhd, \tau, \delta, I, F)$ one can effectively construct an NLAwtl $B = (Q_B, \Sigma, \rhd, \lhd, \tau_B, \delta_B, I_B, F_B)$ in normal form such that $L(B) = L(A)$.*

Proof. Let $A = (Q, \Sigma, \rhd, \lhd, \tau, \delta, I, F)$ be an NLAwtl. We proceed in two steps, corresponding to the two conditions in the definition above.

First we transform A into an NLAwtl $A' = (Q', \Sigma, \rhd, \lhd, \tau', \delta', I', F')$ by choosing a set of additional states $F_2 = \{\, q' \mid q \in F \,\}$ that correspond to the final states of A with respect to the bijection $\beta : F \to F_2$ that maps $q \in F$ to $\beta(q) = q' \in F_2$ and by taking $Q' = Q \cup F_2$, $I' = I \cup \beta(I \cap F)$, and $F' = F_2$. Further, $\tau'(q) = \tau(q)$ for all $q \in Q$ and $\tau'(q') = \emptyset$ for all $q' \in F_2$, and δ' is obtained from δ by adding the following transitions:

$$\delta'(\beta(q), a) \quad = \quad \{\beta(q)\} \quad \text{for all } q \in F \text{ and all } a \in \tau(q).$$

Finally, we take $\delta'(q,a) = \delta(q,a) \cup \{\, \beta(p) \mid p \in \delta(q,a) \cap F \,\}$, that is, if A, while being in state q, can read the letter a and enter the final state $p \in F$, then A' has the additional option to enter state $\beta(p) \in F_2$. Now A' can simulate each computation of A step by step. When A enters a final state q in a situation, where the tape only

contains letters from $\tau(q)$, which means that A will accept, then A' can enter the state $q' = \beta(q)$, read (and remove) all letters on the tape, and accept with empty tape. Thus, we see that A' satisfies the second condition required for an NLAwtl that is in normal form.

Next we transform A' into an NLAwtl $B = (Q_B, \Sigma, \rhd, \lhd, \tau_B, \delta_B, I_B, F_B)$ that is in normal form. Let $q \in Q'$ such that $\mu(q) = \{a_{i_1}, a_{i_2}, \ldots, a_{i_m}\}$ for some $m \geq 2$. Then we replace the state q by m states, say $q_{i_1}, q_{i_2}, \ldots, q_{i_m}$, take $\tau_B(q_{i_j}) = \tau'(q)$ for all $j = 1, 2, \ldots, m$, and define $\delta_B(q_{i_j}, a_{i_j}) = \delta'(q, a_{i_j})$ and $\delta_B(q_{i_j}, b) = \emptyset$ for all $j = 1, 2, \ldots, m$ and all $b \neq a_{i_j}$. Further, in I', in F', and on the right-hand side of δ', we replace each occurrence of q by $q_{i_1}, q_{i_2}, \ldots, q_{i_m}$. Thus, whenever A' enters state q within a computation, then in the corresponding computation of B, one must choose one of the states $q_{i_1}, q_{i_2}, \ldots, q_{i_m}$, in this way guessing which letter a_{i_j} will be read in the following step. It is now obvious that B is in normal form and that it accepts the same language as A. □

If A is an NLAwtl on Σ that is in normal form, then by removing the translucency relation from A, we obtain an NLA A' that accepts a linear sublanguage of $L(A)$. In fact, the following result holds

Proposition 3.6. *By removing the translucency relation from an NLAwtl A that is in normal form, we obtain an NLA A' such that $L(A')$ is a sublanguage of $L(A)$ that is letter-equivalent to $L(A)$.*

Here two languages over the same alphabet $\Sigma = \{a_1, a_2, \ldots, a_n\}$ are called *letter-equivalent* if they have the same image under the Parikh mapping $\psi : \Sigma^* \to \mathbb{N}^n$. Thus, we see that each language from $\mathcal{L}(\mathsf{NLAwtl})$ is letter-equivalent to a linear context-free language and therewith to a regular language.

Proof. Let $A = (Q, \Sigma, \rhd, \lhd, \tau, \delta, I, F)$ be an NLAwtl that is in normal form, and let $B = (Q, \Sigma, \delta, I, F)$ be the NLA that is obtained from A by removing the translucency relation τ and the endmarkers $\rhd$ and $\lhd$. Then each accepting computation of B corresponds to an accepting computation of A in which no translucent letter is ever skipped over. Thus, $L(B) \subseteq L(A)$, that is, $L(B)$ is a sublanguage of $L(A)$.

Conversely, assume that $w \in L(A)$, where $w = a_1 a_2 \cdots a_n$, and that

$$q_0 \rhd w \lhd \vdash_A q_{i_1} \rhd w_1 \lhd \vdash_A q_{i_2} \rhd w_2 \lhd \vdash_A \cdots \vdash_A q_{i_{n-1}} \rhd w_{n-1} \lhd \vdash_A q_{i_n} \rhd \lhd$$

is an accepting computation of A on input w. Then $q_0 \in I$ and $q_{i_n} \in F$. We claim that there exists a word $z \in \Sigma^*$ such that $q_0 z \vdash_B^n q_{i_n}$ and $\psi(z) = \psi(w)$ hold. We proceed by induction on the length n of w.

If $n = 0$, then $w = \varepsilon$, which means that $q_0 = q_{i_n}$. Hence, we can take $z = \varepsilon = w$.

Now assume that $q_0 \in Q_L$ and that $w = xay$ for some $x \in (\tau(q_0))^*$ and $a \in \Sigma \setminus \tau(q_0)$. Then $w_1 = xy$ and $q_{i_1} \in \delta(q_0, a)$. From the induction hypothesis we see that there exists a word $z_1 \in \Sigma^*$ such that $q_{i_1} z_1 \vdash_B^{n-1} q_{i_n}$ and $\psi(z_1) = \psi(w_1)$. Let $z = a z_1$. Then $q_0 z = q_0 a z_1 \vdash_B q_{i_1} z_1 \vdash_B^{n-1} q_{i_n}$, that is, B accepts on input z, and $\psi(z) = \psi(a z_1) = \psi(a w_1) = \psi(axy) = \psi(xay) = \psi(w)$.

Finally, if $q_0 \in Q_R$ and $w = ubv$ for some $v \in (\tau(q_0))^*$ and $b \in \Sigma \setminus \tau(q_0)$, then $w_1 = uv$ and $q_{i_1} \in \delta(q_0, b)$. Again from the induction hypothesis we see that there exists a word $z_1 \in \Sigma^*$ such that $q_{i_1} z_1 \vdash_B^{n-1} q_{i_n}$ and $\psi(z_1) = \psi(w_1)$. Let $z = z_1 b$. Then $q_0 z = q_0 z_1 b \vdash_B q_{i_1} z_1 \vdash_B^{n-1} q_{i_n}$, that is, B accepts on input z, and $\psi(z) = \psi(z_1 b) = \psi(w_1 b) = \psi(uvb) = \psi(ubv) = \psi(w)$. Thus, we see that, for each word $w \in L(A)$, there exists a word $z \in L(B)$ such that w and z are letter-equivalent. □

As NLAs accept exactly the linear context-free languages, this yields the following consequence.

Corollary 3.7. *Each language $L \in \mathcal{L}(\mathsf{NLAwtl})$ contains a sublanguage that is linear context-free and that is letter-equivalent to L. In particular, this implies that L is semi-linear, that is, $\psi(L)$ is a semi-linear subset of $\mathbb{N}^n$.*

As an example, we take the language

$$L' = \{\, a^m b^n \# u \# a^m b^n \mid m, n \geq 0, u \in \{c, d\}^* \text{ is a palindrome} \,\},$$

which is a sublanguage of the language L considered in Example 3.2. The language L' does not contain a linear sublanguage that is letter-equivalent to the language itself, and so it follows from Corollary 3.7 that this language is not accepted by any NLAwtl.

4. Linear context-free trace languages

Let Σ be a finite alphabet, and let D be a binary relation on Σ that is reflexive and symmetric, that is, $(a, a) \in D$ for all $a \in \Sigma$, and $(a, b) \in D$ implies that $(b, a) \in D$, too. Then D is called a *dependency relation* on Σ, and the relation $I_D = (\Sigma \times \Sigma) \setminus D$ is called the corresponding *independence relation*. Obviously, the relation I_D is irreflexive and symmetric. The independence relation I_D induces a binary relation $\equiv_D$ on Σ^* that is defined as the smallest congruence relation containing the set of pairs $\{\, (ab, ba) \mid (a, b) \in I_D \,\}$. For $w \in \Sigma^*$, the congruence class of w mod $\equiv_D$ is denoted by $[w]_D$, that is, $[w]_D = \{\, z \in \Sigma^* \mid w \equiv_D z \,\}$. These congruence classes are called *traces*, and the factor monoid $M(D) = \Sigma^*/\equiv_D$ is a *trace monoid*. In fact, $M(D)$ is the *free partially commutative monoid* presented by (Σ, D) (see, *e.g.*, [8]).

Traces are being used in concurrency theory to describe sequences of actions that are partially independent of each other. Let a and b symbolize two actions that are executed in parallel. If they are independent, then in a sequential simulation it does not matter in which order they are executed, that is, ab and ba are equivalent. As such traces have received much attention (see, *e.g.*, [8]).

To simplify the notation in what follows, we introduce the following notions. For $w \in \Sigma^*$, we use $\mathrm{Alph}(w)$ to denote the set of all letters that occur in w, that is,

$$\mathrm{Alph}(w) = \{\, a \in \Sigma \mid |w|_a > 0 \,\}.$$

Now we extend the independence relation from letters to words by defining, for all words $u, v \in \Sigma^*$,

$$(u, v) \in I_D \text{ if and only if } \mathrm{Alph}(u) \times \mathrm{Alph}(v) \subseteq I_D.$$

As $\mathrm{Alph}(\varepsilon) = \emptyset$, we see that $(\varepsilon, w) \in I_D$ for every word $w \in \Sigma^*$. The following technical result (see, *e.g.*, Claim A in the proof of Prop. 6.2.2 of [8]) will be useful in what follows.

Proposition 4.1. *For all words $x, y, u \in \Sigma^*$ and all letters $a \in \Sigma$, if $xay \equiv_D au$ and $|x|_a = 0$, then $(a, x) \in I_D$, $xay \equiv_D axy$, and $xy \equiv_D u$.*

A subset S of a trace monoid $M(D)$ is called *recognizable* if there exist a finite monoid N, a morphism $\alpha : M(D) \to N$, and a subset P of N such that $S = \alpha^{-1}(P)$ [3]. Accordingly, this property can be characterized as follows (see Prop. 6.1.10 of [8]).

Proposition 4.2. *Let $M(D)$ be the trace monoid presented by (Σ, D), and let $\varphi_D : \Sigma^* \to M(D)$ be the corresponding morphism. Then a set $S \subseteq M(D)$ is recognizable if and only if the language $\varphi_D^{-1}(S)$ is a regular language over Σ.*

By $\mathsf{REC}(M(D))$ we denote the set of recognizable subsets of $M(D)$.

A subset S of a trace monoid $M(D)$ is called *rational* if it can be obtained from singleton sets by a finite number of unions, products, and star operations [3]. This property can be characterized more conveniently as follows.

Proposition 4.3. *Let $M(D)$ be the trace monoid presented by (Σ, D), and let $\varphi_D : \Sigma^* \to M(D)$ be the corresponding morphism. Then a set $S \subseteq M(D)$ is rational if and only if there exists a regular language R over Σ such that $S = \varphi_D(R)$.*

By $\mathsf{RAT}(M(D))$ we denote the set of rational subsets of $M(D)$. Concerning the relationship between the recognizable subsets of $M(D)$ and the rational subsets of $M(D)$ the following results are known (see, *e.g.*, [8]).

Proposition 4.4. *For each trace monoid $M(D)$, $\mathsf{REC}(M(D)) \subseteq \mathsf{RAT}(M(D))$, and these two sets are equal if and only if $I_D = \emptyset$.*

Thus, each recognizable subset of a trace monoid $M(D)$ is necessarily rational, but the converse only holds if I_D is empty, that is, if $D = \Sigma \times \Sigma$, which means that the congruence $\equiv_D$ is the identity. Thus, the free monoids are the only trace monoids for which the recognizable subsets coincide with the rational subsets.

We call a language $L \subseteq \Sigma^*$ a *rational trace language* if there exists a dependency relation D on Σ such that $L = \varphi_D^{-1}(S)$ for a rational subset S of the trace monoid $M(D)$ presented by (Σ, D). From Proposition 4.3 it follows that L is a rational trace language if and only if there exist a trace monoid $M(D)$ and a regular language $R \subseteq \Sigma^*$ such that $L = \varphi_D^{-1}(\varphi_D(R)) = \bigcup_{w \in R}[w]_D$. By $\mathcal{LRAT}(D)$ we denote the set of rational trace languages $\varphi_D^{-1}(\mathsf{RAT}(M(D)))$, and $\mathcal{LRAT}$ is the class of all rational trace languages. In [22] (see also [25]) the following result on rational trace languages was established.

Theorem 4.5. *$\mathcal{LRAT} \subsetneq \mathcal{L}(\mathsf{NFAwtl})$, that is, if $M(D)$ be the trace monoid presented by (Σ, D), where D is a dependency relation on the finite alphabet Σ, then the language $\varphi_D^{-1}(S)$ is accepted by an NFAwtl for each rational set of traces $S \subseteq M(D)$.*

Here we are interested in more general trace languages. A language $L \subseteq \Sigma^*$ is called a *linear context-free trace language* if there exist a dependency relation D on Σ and a linear context-free language $R \subseteq \Sigma^*$ such that $L = \varphi_D^{-1}(\varphi_D(R)) = \bigcup_{w \in R}[w]_D$. Analogously, a language $L \subseteq \Sigma^*$ is called a *context-free trace language* if there exist a dependency relation D on Σ and a context-free language $R \subseteq \Sigma^*$ such that $L = \varphi_D^{-1}(\varphi_D(R)) = \bigcup_{w \in R}[w]_D$ [1, 4]. By $\mathcal{LLCF}(D)$ we denote the set of linear context-free trace languages obtained from (Σ, D), and $\mathcal{LLCF}$ is the class of all linear context-free trace languages. Further, by $\mathcal{LCF}(D)$ we denote the set of context-free trace languages obtained from (Σ, D), and $\mathcal{LCF}$ is the class of all context-free trace languages. In [23, 24] it has been shown that the context-free trace languages are accepted by certain cooperating distributed systems of a very restricted type of restarting automata. These systems can actually be interpreted as nondeterministic pushdown automata with translucent letters. Here we derive the following result for linear context-free trace languages.

Theorem 4.6. *Let $M(D)$ be the trace monoid presented by (Σ, D), where D is a dependency relation on the finite alphabet Σ. Then*

$$\mathcal{LLCF}(D) \subseteq \mathcal{L}(\mathsf{NLAwtl}),$$

that is, the language $\varphi_D^{-1}(\varphi_D(R))$ is accepted by an NLAwtl for each linear context-free language $R \subseteq \Sigma^*$.

Proof. Let R be a linear context-free language over Σ, let $S = \varphi_D(R) \subseteq M(D)$, and let $L = \varphi_D^{-1}(S) \subseteq \Sigma^*$ be the linear context-free trace language defined by R and D. As $R \subseteq \Sigma^*$ is a linear context-free language, there exists an NLA $A = (Q, \Sigma, \delta, I, F)$ such that $L(A) = R$. According to our remarks on NLAs in Section 2, we can assume without loss of generality that the NLA A is in normal form, that is, for each state $q \in Q$, there is at most a single letter $a_q \in \Sigma$ such that $\delta(q, a_q) \neq \emptyset$. Further, based on well-known transformation techniques (see, *e.g.*, [15, 17]), we may assume that F consists of a single left state q_+ only, and that $\delta(q_+, a) = \emptyset$ for all letters $a \in \Sigma$ (that is, no transition is allowed from the accepting state). From A we now construct an NLAwtl $B = (Q_B, \Sigma, \rhd, \lhd, \tau, \delta_B, I_B, F_B)$ as follows:

- $Q_B = Q$, $I_B = I$, and $F_B = F = \{q_+\}$,

– $\tau(q) = \{\, b \in \Sigma \mid (b, a_q) \in I_D \,\}$ for all $q \in Q \setminus F$, that is, all those letters are translucent for state q that are independent of the letter a_q that A can read (and delete) in state q, and $\tau(q_+) = \emptyset$, and
– the transition relation δ_B is simply defined by taking $\delta_B(q, a) = \delta(q, a)$ for all $q \in Q$ and all $a \in \Sigma$.

It remains to show that $L(B) = L = \varphi_D^{-1}(S) = \bigcup_{u \in R}[u]_D$. Notice that in order to accept a word w, B must read (and delete) it completely, as $\tau(q_+) = \emptyset$ and q_+ is the only final state of B.

Claim 1. $\bigcup_{u \in R}[u]_D \subseteq L(B)$.

Proof. Assume that $w \in \bigcup_{u \in R}[u]_D$. Then there exists a word $u \in R$ such that $w \equiv_D u$, and so there exists a sequence of words $u = w_0, w_1, \ldots, w_n = w$ such that, for each $i = 1, 2, \ldots, n$, w_i is obtained from w_{i-1} by replacing a factor ab by ba for some pair of letters $(a, b) \in I_D$. We now prove that $w_i \in L(B)$ for all i by induction on i.

For $i = 0$ we have $w_0 = u \in R$. Thus, w_0 is accepted by the NLA A, and it follows immediately from the definition of B that w_0 is also accepted by B.

Now assume that $w_i \in L(B)$ for some $i \geq 0$, and that $w_i = xaby$ and $w_{i+1} = xbay$ for a pair of letters $(a, b) \in I_D$. By our hypothesis B has an accepting computation for $w_i = xaby$, which is of one of the following two forms:

$$(1)\ q_1 \rhd w_i \lhd = q_1 \rhd xaby \lhd \vdash_B^m q_2 \rhd x'aby' \lhd \vdash_B q_3 \rhd x'by' \lhd \vdash_B^* q_+ \rhd \lhd,$$

or

$$(2)\ q_1 \rhd w_i \lhd = q_1 \rhd xaby \lhd \vdash_B^m q_2 \rhd x'aby' \lhd \vdash_B q_3 \rhd x'ay' \lhd \vdash_B^* q_+ \rhd \lhd,$$

where in the first m steps some letters from x and y are read and deleted, in this way reducing these factors to x' and y', respectively, and $q_3 \in \delta_B(q_2, a)$ (in (1)) or $q_3 \in \delta_B(q_2, b)$ (in (2)). Now consider the computation of B on input $w_{i+1} = xbay$. In (1), if q_2 is a right state, then obviously we have the computation

$$q_1 \rhd w_{i+1} \lhd = q_1 \rhd xbay \lhd \vdash_B^m q_2 \rhd x'bay' \lhd \vdash_B q_3 \rhd x'by' \lhd \vdash_B^* q_+ \rhd \lhd.$$

If q_2 is a left state, then observe that $(b, a) \in I_D$, and hence, we see that $b \in \tau(q_2)$, and so also in this situation B can execute the computation

$$q_1 \rhd w_{i+1} \lhd = q_1 \rhd xbay \lhd \vdash_B^m q_2 \rhd x'bay' \lhd \vdash_B q_3 \rhd x'by' \lhd \vdash_B^* q_+ \rhd \lhd.$$

Case (2) is obviously symmetric to (1). Thus, we see that $w_{i+1} \in L(B)$. This completes the proof of Claim 1. □

Claim 2. $L(B) \subseteq \bigcup_{u \in R}[u]_D$.

Proof. Let $w \in L(B)$, and let

$$\begin{array}{ccccccc} q_n \rhd w \lhd = q_n \rhd w_n \lhd & \vdash_B & q_{n-1} \rhd w_{n-1} \lhd & \vdash_B & q_{n-2} \rhd w_{n-2} \lhd & \vdash_B \\ \cdots & \vdash_B & q_1 \rhd w_1 \lhd & \vdash_B & q_+ \rhd \lhd & \end{array}$$

be an accepting computation of B on input w, where $q_n \in I_B$. We claim that, for each $i = 1, 2, \ldots, n$, there exists a word $u_i \in \Sigma^*$ such that $u_i \equiv_D w_i$ and $q_i u_i \vdash_A^* q_+$, that is, the NLA A accepts the word u_i when starting from state q_i.

We prove this claim by induction on i. For $i = 1$ we have $w_i = a_1$, and $q_+ \in \delta_B(q_1, a_1)$. From the definition of B we see that $q_+ \in \delta(q_1, a_1)$, and so we can simply take $u_1 = a_1 = w_1$. Now assume that, for some $i \geq 1$, $u_i \equiv_D w_i$ and $q_i u_i \vdash_A^* q_+$ hold. The above computation of B contains the step $q_{i+1} \rhd w_{i+1} \lhd \vdash_B q_i \rhd w_i \lhd$. Again

from the definition of B we see that either q_{i+1} is a left state and $q_i \in \delta(q_{i+1}, a_{i+1})$, where $w_{i+1} = xa_{i+1}y$ and $w_i = xy$ for some words $x, y \in \Sigma^*$ such that $(x, a_{i+1}) \in I_D$, or q_{i+1} is a right state and $q_i \in \delta(q_{i+1}, a_{i+1})$ with $w_{i+1} = xa_{i+1}y$ and $w_i = xy$ for some words $x, y \in \Sigma^*$ such that $(y, a_{i+1}) \in I_D$. In the former case let u_{i+1} be the word $u_{i+1} = a_{i+1}u_i$, and in the latter case, let u_{i+1} be the word $u_{i+1} = u_i a_{i+1}$. Then

$$u_{i+1} = a_{i+1}u_i \equiv_D a_{i+1}w_i = a_{i+1}xy \equiv_D xa_{i+1}y = w_{i+1}$$

and $q_{i+1}u_{i+1} = q_{i+1}a_{i+1}u_i \vdash_A q_i u_i$, or

$$u_{i+1} = u_i a_{i+1} \equiv_D w_i a_{i+1} = xya_{i+1} \equiv_D xa_{i+1}y = w_{i+1}$$

and $q_{i+1}u_{i+1} = q_{i+1}u_i a_{i+1} \vdash_A q_i u_i$. As by the induction hypothesis $q_i u_i \vdash_A^* q_+$, we see that in either case $q_{i+1}u_{i+1} \vdash_A^* q_+$ follows.

For $i = n$ we obtain a word $u \in \Sigma^*$ such that $u \equiv_D w$, and A accepts u starting from state $q_n \in I$. Hence, $u \in R$, and it follows that $L(B) \subseteq \bigcup_{u \in R}[u]_D$ holds. □

Now Claims 1 and 2 together show that $L(B) = \bigcup_{u \in R}[u]_D = \varphi_D^{-1}(S) = L$, which completes the proof of Theorem 4.6. □

Next we present a restricted class of NLAwtls that accept exactly the linear context-free trace languages.

Definition 4.7. Let $B = (Q, \Sigma, \rhd, \lhd, \tau, \delta, I, F)$ be an NLAwtl in normal form that satisfies the following additional condition:

$$(*) \quad \forall p, q \in Q : \mu(p) = \mu(q) \text{ implies that } \tau(p) = \tau(q),$$

that is, if two states read (and delete) the same letter, then they also have the same set of translucent letters. With B we associate a binary relation

$$I_B = \bigcup_{q \in Q} (\tau(q) \times \mu(q)),$$

that is, $(a, b) \in I_B$ if and only if there exists a state $q \in Q$ such that $a \in \tau(q)$ and $\delta(q, b)$ is defined. The automaton B is called a $\tau\mu$-NLAwtl, and further, by D_B we denote the relation $D_B = (\Sigma \times \Sigma) \setminus I_B$.

Observe that the relation I_B defined above is necessarily irreflexive, but that it will in general not be symmetric. On the other hand, consider the NLAwtl B that is constructed in the proof of Theorem 4.6. It satisfies condition $(*)$ above and the relation I_B coincides with the relation I_D restricted to the subset of letters of Σ that actually occur in the language L. Hence, this relation is in fact symmetric. The following result shows that also the converse of this observation holds.

Theorem 4.8. *Let B be an NLAwtl over Σ satisfying condition $(*)$ above, that is, a $\tau\mu$-NLAwtl. If the associated relation I_B is symmetric, then $L(B)$ is a linear context-free trace language over Σ. In fact, from B one can construct an NLA A over Σ such that $L(B) = \varphi_{D_B}^{-1}(\varphi_{D_B}(L(A)))$.*

Proof. Let $B = (Q, \Sigma, \rhd, \lhd, \tau, \delta, I, F)$ be a $\tau\mu$-NLAwtl and assume that the associated relation $I_B = \bigcup_{q \in Q}(\tau(q) \times \mu(q))$ is symmetric. Then the relation $D_B = (\Sigma \times \Sigma) \setminus I_B$ is reflexive and symmetric, and so it is a dependency relation on Σ with associated independence relation I_B. Without loss of generality we may assume that all letters from Σ do actually occur in some words of $L(B)$, since otherwise we could simply remove these letters from Σ. From the properties of B we obtain the following consequences:

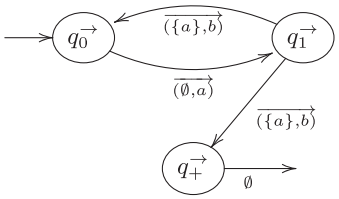


FIGURE 2. The graphical representation of the NLAwtl A for the Dyck language D_1.

1. As all words $w \in L(B)$ are first reduced to the empty word, which is then accepted in a final state, we see that, for each letter $b \in \Sigma$, there exists a state $q \in Q$ such that $\mu(q) = \{b\}$.
2. If $(a, b) \in I_B$, then $a \in \tau(q)$ for all states q for which $\mu(q) = \{b\}$ holds (by $(*)$).
3. If $(a, b) \in I_B$, then $(b, a) \in I_B$, too, and hence, $b \in \tau(p)$ for all states p for which $\mu(p) = \{a\}$ holds (by symmetry of I_B and by $(*)$).

Let $L = L(B)$. We claim that L is a linear context-free trace language over the trace monoid defined by (Σ, D_B), that is, $L \in \mathcal{LLCF}(D_B)$. To verify this claim we present a linear context-free language $R \subseteq \Sigma^*$ such that $L = \bigcup_{z \in R} [z]_{D_B}$.

By Proposition 3.6, the language L contains a linear context-free sublanguage R that is letter-equivalent to L. In the proof of that proposition, an NLA $A = (Q_A, \Sigma, \delta_A, I_A, F_A)$ for R is constructed from B by simply removing the translucency relation. In addition, it is shown that each word $w \in L$ is letter-equivalent to a word $z \in R$. Actually, it follows that $w \equiv_{D_B} z$, which implies that $L \subseteq \bigcup_{z \in R} [z]_{D_B}$.

Conversely, let $w \equiv_{D_B} u \in R$, and let $u = w_0, w_1, \ldots, w_n = w$ be a sequence of words such that, for each $i = 1, 2, \ldots, n$, w_i is obtained from w_{i-1} by replacing a factor ab by ba for some pair of letters $(a, b) \in I_B$. We now prove that $w_i \in L$ for all i by induction on i.

For $i = 0$ we have $w_0 = u \in R$, and so $w_0 \in L$, as R is a sublanguage of L. Now assume that $w_i \in L$ for some $i \geq 0$, and that $w_i = xaby$ and $w_{i+1} = xbay$ for a pair of letters $(a, b) \in I_B$. By our hypothesis, B has an accepting computation for $w_i = xaby$. As $a \in \tau(q)$ for each state q satisfying $\mu(q) = \{b\}$ and $b \in \tau(p)$ for each state p satisfying $\mu(p) = \{a\}$, we see that the same computation of B applies to the word $w_{i+1} = xbay$, which shows that $w_{i+1} \in L$. Thus, we see that $w = w_n$ is accepted by B. This implies that $L = \bigcup_{z \in R} [z]_{D_B}$, that is, L is a linear context-free trace language. □

Theorems 4.6 and 4.8 together yield the following characterization.

Corollary 4.9. *A language $L \subseteq \Sigma^*$ is a linear context-free trace language if and only if there exists a $\tau\mu$-NLAwtl B with symmetric relation I_B such that $L = L(B)$.*

Let D_1 be the Dyck language over $\Sigma = \{a, b\}$, that is, D_1 is the language generated by the context-free grammar $(\{S\}, \{a, b\}, S, \{S \to SS, S \to aSb, S \to \varepsilon\})$. This language is accepted by the NLAwtl A that is given through the diagram in Figure 2.

Actually, A only uses its left head, that is, it is really just an NFAwtl. This automaton satisfies condition $(*)$ above, but the relation $I_A = \tau(q_1) \times \mu(q_1) = \{(a, b)\}$ is not symmetric. And indeed, the language D_1 is not a linear context-free trace language.

5. Closure properties and decidability results

While the language

$$L = \{ w_1 \# u \# w_2 \mid w_1, w_2 \in \{a, b\}^*, |w_1|_a = |w_2|_a, |w_1|_b = |w_2|_b, \text{ and } u \in \{c, d\}^* \text{ is a palindrome} \}$$

is accepted by a DLAwtl (see Ex. 3.2), it follows from Corollary 3.7 that the language

$$\begin{aligned} L' &= L \cap a^* \cdot b^* \cdot \# \cdot \{c,d\}^* \cdot \# \cdot a^* \cdot b^* \\ &= \{\, a^m b^n \# u \# a^m b^n \mid m,n \geq 0,\, u \in \{c,d\}^* \text{ is a palindrome} \,\} \end{aligned}$$

is not accepted by any NLAwtl. This observation implies that the language classes $\mathcal{L}(\mathsf{NLAwtl})$ and $\mathcal{L}(\mathsf{DLAwtl})$ are not closed under intersection with regular languages. As all regular languages are accepted by DLAwtls, this means that these classes are not closed under intersection, either. However, it is easily seen that the former class is closed under union. Thus, we have the following consequence.

Corollary 5.1. *The language class $\mathcal{L}(\mathsf{NLAwtl})$ is closed under union, but it is neither closed under intersection nor under complementation.*

Concerning DLAwtls, we have the following closure property.

Proposition 5.2. *The language class $\mathcal{L}(\mathsf{DLAwtl})$ is closed under complementation.*

Proof. Let $A = (Q, \Sigma, \rhd, \lhd, \tau, \delta, q_0, F)$ be a DLAwtl. We define a DLAwtl $A^c = (Q \cup \{q_f\}, \Sigma, \rhd, \lhd, \tau_c, \delta_c, q_0, F_c)$, where q_f is a new left state, by taking $\tau_c(q_f) = \Sigma$ and $\tau_c(q) = \tau(q)$ for all $q \in Q$, $F_c = (Q \smallsetminus F) \cup \{q_f\}$, and by defining δ_c as follows:

$$\begin{aligned} \delta_c(q,a) &= \delta(q,a) && \text{for all } q \in Q \text{ and all } a \in \mu(q), \\ \delta_c(q,b) &= q_f && \text{for all } q \in Q \text{ and all } b \in \Sigma \smallsetminus (\tau(q) \cup \mu(q)). \end{aligned}$$

Thus, given a word $w \in \Sigma^*$ as input, A^c performs the same steps as A until A halts. Let $q \in Q$ and $u \in \Sigma^*$ be the current state and the current tape contents at that point.

- If $q \in F$ and $u \in (\tau(q))^*$, then A accepts, which means that $w \in L(A)$. However, as $q \notin F_c$, we see that A^c does not accept on input w.
- If $q \notin F$, but $u \in (\tau(q))^*$, then A does not accept, which means that $w \notin L(A)$. However, as $q \in F_c$, we see that A^c accepts on input w.
- Finally, if q is a left state and $u = xay$ for some word $x \in (\tau(q))^*$ and a letter $a \notin (\tau(q) \cup \mu(q))$, then A just gets stuck without accepting, which means that $w \notin L(A)$. However, A^c continues the current computation by $q \rhd xay\lhd \vdash_{A^c} q_f \rhd xy\lhd$, and then A^c accepts as $q_f \in F_c$ and $xy \in (\tau(q_f))^*$, that is, $w \in L(A^c)$.
- If q is a right state and $u = xay$ for some word $y \in (\tau(q))^*$ and a letter $a \notin (\tau(q) \cup \mu(q))$, then it follows analogously that A^c accepts on input w.

Thus, we see that $L(A^c) = \Sigma^* \smallsetminus L(A)$. □

As observed above, the language class $\mathcal{L}(\mathsf{DLAwtl})$ is not closed under intersection with regular sets. Thus, we get the following consequence from Proposition 5.2.

Corollary 5.3. *The language class $\mathcal{L}(\mathsf{DLAwtl})$ is not closed under intersection with regular sets, and hence, it is neither closed under union nor under intersection.*

From the definition it follows immediately that $\mathcal{L}(\mathsf{DLAwtl}) \subseteq \mathcal{L}(\mathsf{NLAwtl})$. By Proposition 5.2, $\mathcal{L}(\mathsf{DLAwtl})$ is closed under complementation, while by Corollary 5.1, $\mathcal{L}(\mathsf{NLAwtl})$ is not. Thus, we obtain the following separation result.

Corollary 5.4. $\mathcal{L}(\mathsf{DLAwtl}) \subsetneq \mathcal{L}(\mathsf{NLAwtl})$.

In fact, in Proposition 5.7 below, we will encounter an example language that separates the class $\mathcal{L}(\mathsf{DLAwtl})$ from the class $\mathcal{L}(\mathsf{NLAwtl})$.

The languages $L_1 = \{ a^m b^m \mid m \geq 0 \}$ and $L_2 = \{ c^n d^n \mid n \geq 0 \}$ are both linear context-free and it is easily seen they are accepted by DLAwtls. However, their product $L_1 \cdot L_2 = \{ a^m b^m c^n d^n \mid m, n \geq 0 \}$ is not linear context-free, and in fact it does not even contain a sublanguage that is linear context-free and letter-equivalent to the language itself. Hence, by Corollary 3.7, $L_1 \cdot L_2$ is not accepted by any NLAwtl. This yields the following non-closure property.

Corollary 5.5. *The language classes* $\mathcal{L}(\mathsf{NLAwtl})$ *and* $\mathcal{L}(\mathsf{DLAwtl})$ *are not closed under product.*

From the definition of the NLAwtl, it follows immediately that the language classes $\mathcal{L}(\mathsf{NLAwtl})$ and $\mathcal{L}(\mathsf{DLAwtl})$ are closed under reversal. In addition, the former class is closed under a weaker product operation.

Proposition 5.6. *The language class* $\mathcal{L}(\mathsf{NLAwtl})$ *is closed under product with regular sets, that is, if* $L \in \mathcal{L}(\mathsf{NLAwtl})$ *and* $R \in \mathsf{REG}$*, then* $L \cdot R$ *and* $R \cdot L$ *are accepted by NLAwtls.*

Proof. Let $A = (Q, \Sigma, \rhd, \lhd, \tau, \delta, I, F)$ be an NLAwtl in normal form, and let $B = (Q', \Sigma, \delta', q_0', F')$ be an NFA. From A and B we obtain an NLAwtl C for the product $L(B) \cdot L(A)$ as follows:

- First C simulates the NFA B on a prefix u of the given input w. Whenever B enters a final state from F', then C has the option of entering an initial state of A.
- Once C has entered an initial state of A, it simulates A on the remaining suffix v of the input w. C accepts if and when A accepts.

Thus, C accepts a given input w iff w admits a factorization of the form $w = uv$ such that $u \in L(B)$ and $v \in L(A)$, which shows that $L(C) = L(B) \cdot L(A)$. As $L(A) \cdot L(B) = (L(B)^R \cdot L(A)^R)^R$, and as REG as well as $\mathcal{L}(\mathsf{NLAwtl})$ are both closed under reversal, we see that also $L(A) \cdot L(B)$ is accepted by an NLAwtl. □

In [27] it has been noted that 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[2]. As $L_\vee$ is linear context-free, it is accepted by an NLAwtl. However, generalizing the proof of Proposition 4.7 of [27], we now show that $L_\vee$ is not accepted by any DLAwtl. Thus, $L_\vee$ separates the class $\mathcal{L}(\mathsf{DLAwtl})$ from the class $\mathcal{L}(\mathsf{NLAwtl})$.

Proposition 5.7. *The language* $L_\vee$ *is not accepted by any DLAwtl.*

Proof. Assume that $A = (Q, \{a, b\}, \rhd, \lhd, \tau, \delta, \{q_0\}, F)$ is a DLAwtl that accepts the language $L_\vee$. Let m be a sufficiently large integer, and let $w = a^m b^m$. Then $w \in L_\vee$, and hence, the computation of A on input w is accepting. Let

$$q_0 \rhd w \lhd = q_0 \rhd a^m b^m \lhd \vdash_A^* q_f \rhd x \lhd$$

be this computation, where $q_f \in F$ and $x \in (\tau(q_f))^*$. As x is obtained from $w = a^m b^m$ by deleting some letters, we see that $x = a^r b^s$ for some integers r and s such that $0 \leq r, s \leq m$. If $r > 0$, then $a \in \tau(q_f)$, and then A would also accept the input $a^{m+1} b^m \notin L_\vee$. Analogously, if $s > 0$, then $b \in \tau(q_f)$, and then A would also accept the input $a^m b^{m+1} \notin L_\vee$. Hence, we conclude that $r = s = 0$, that is, $x = \varepsilon$, which implies that the accepting computation of A on input w consists of $2m$ steps.

[2] Actually, it is shown in [27] that $L_\vee$ is not accepted by any stl-det-global-CD-R(1)-system, which is a specific type of cooperating distributed system of a very restricted type of restarting automata. The results of [25] then yield that $L_\vee$ is not accepted by any DFAwtl.

If $m > |Q|$, then there exists a state $q \in Q$ that occurs repeatedly in this computation, that is, this computation can be written as follows:

$$q_0 \rhd w \lhd = q_0 \rhd a^m b^m \lhd \vdash_A^i q \rhd a^{m-i_1} b^{m-i_2} \lhd \vdash_A^{j-i} q \rhd a^{m-j_1} b^{m-j_2} \lhd \vdash_A^* q_f \rhd \lhd,$$

where $0 \leq i < j \leq |Q|$, $i_1, i_2, j_1, j_2 \in \mathbb{N}$ such that $i = i_1 + i_2$ and $j = j_1 + j_2$, and $q_f \in F$. Of course, $j_1 \geq i_1$ and $j_2 \geq i_2$.

Claim. $j_1 - i_1 = j_2 - i_2$.

Proof. Assume that $j_1 - i_1 > j_2 - i_2$. Then $m + j_1 - i_1 > m + j_2 - i_2$, and hence, the word $a^{m+j_1-i_1} b^{m+j_2-i_2}$ is not an element of $L_\vee$. However, on input $a^{m+j_1-i_1} b^{m+j_2-i_2}$, A executes the following computation, as A is deterministic:

$$q_0 \rhd a^{m+j_1-i_1} b^{m+j_2-i_2} \lhd \vdash_A^j q \rhd a^{m-i_1} b^{m-i_2} \lhd \vdash_A^* q_f \rhd \lhd,$$

that is, A accepts this word. This contradiction implies that $j_1 - i_1 \leq j_2 - i_2$.

Now assume that $j_1 - i_1 < j_2 - i_2$. Then $m + j_1 - i_1 < m + j_2 - i_2$, and if m is sufficiently large, then $2(m + j_1 - i_1) > m + j_2 - i_2$, and hence, also in this case the word $a^{m+j_1-i_1} b^{m+j_2-i_2}$ is not an element of $L_\vee$. However, as above it follows that A accepts this word. Thus, we see that indeed $j_1 - i_1 = j_2 - i_2$ holds. □

Now we consider the word $z = a^m b^{2m} \in L_\vee$. Then the computation of A on input z is accepting, and as A is deterministic, this computation has the following form:

$$q_0 \rhd z \lhd = q_0 \rhd a^m b^{2m} \lhd \vdash_A^i q \rhd a^{m-i_1} b^{2m-i_2} \lhd \vdash_A^{j-i} q \rhd a^{m-j_1} b^{2m-j_2} \lhd \vdash_A^* p_f \rhd \lhd$$

for some final state $p_f \in F$. From this computation we now obtain the following computation:

$$q_0 \rhd a^{m+j_1-i_1} b^{2m+j_2-i_2} \lhd \vdash_A^j q \rhd a^{m-i_1} b^{2m-i_2} \lhd \vdash_A^* p_f \rhd \lhd.$$

However, $m + j_1 - i_1 \neq 2m + j_1 - i_1 = 2m + j_2 - i_2$ and $2(m + j_1 - i_1) \neq 2m + j_1 - i_1 = 2m + j_2 - i_2$, which implies that the word $a^{m+j_1-i_1} b^{2m+j_2-i_2}$ is not an element of $L_\vee$. Thus, it follows that the language $L_\vee$ is not accepted by any DLAwtl. □

This observation yields the following result.

Corollary 5.8. *The language class* $\mathcal{L}(\mathsf{DLAwtl})$ *does not even contain all rational trace languages.*

Let $L'_\vee$ denote the following variant of the language $L_\vee$:

$$L'_\vee = \{\, wc \mid w \in \{a,b\}^+, |w|_a = |w|_b - 1 \,\} \cup \{\, wd \mid w \in \{a,b\}^+, 2 \cdot |w|_a = |w|_b - 1 \,\}.$$

It is easy to design a DLAwtl A for this language. Now let $\varphi : \{a,b,c,d\}^* \to \{a,b\}^*$ be the alphabetic morphism that is defined by $a \mapsto a$ and $b, c, d \mapsto b$. Then

$$\varphi(L'_\vee) = \{\, w \in \{a,b\}^+ \mid \exists\, n \geq 0 : |w|_a = n \text{ and } |w|_b \in \{n, 2n\} \,\},$$

which is not accepted by any DLAwtl (see the proof of Prop. 5.7). Thus, we obtain the following non-closure property.

Corollary 5.9. *The language class* $\mathcal{L}(\mathsf{DLAwtl})$ *is not closed under alphabetic morphisms.*

However, the following questions remain open.

Question 1: Is any of the classes $\mathcal{L}(\mathsf{DLAwtl})$ and $\mathcal{L}(\mathsf{NLAwtl})$ closed under Kleene star or inverse morphisms, and is $\mathcal{L}(\mathsf{NLAwtl})$ closed under ε-free morphisms?

Consider the linear context-free language $L = \{ a^m b^m \mid m \geq 1 \}$. It is easily seen that L is accepted by a DLAwtl. We think that L^* is not accepted by any NLAwtl, which would prove non-closure under Kleene star.

Finally, we turn to decision problems. A DLAwtl (NLAwtl) can easily be simulated by a (non-) deterministic one-tape Turing machine that is simultaneously linearly space-bounded and quadratically time-bounded. Hence, we have the following complexity results.

Proposition 5.10.

(a) $\mathcal{L}(\mathsf{DLAwtl}) \subseteq \mathsf{DSpaceTime}(n, n^2)$.
(b) $\mathcal{L}(\mathsf{NLAwtl}) \subseteq \mathsf{NSpaceTime}(n, n^2)$.

It follows in particular that $\mathcal{L}(\mathsf{NLAwtl})$ is contained in the class of context-sensitive languages. The proof of Proposition 3.6 yields an effective construction of an NLA A' from an NLAwtl A such that the language $E = L(A')$ is a subset of the language $L = L(A)$ that is letter-equivalent to L. Hence, E is non-empty if and only if L is non-empty, and E is infinite if and only if L is infinite. As the emptiness problem and the finiteness problem are decidable for NLAs (that is, for linear context-free languages), this immediately yields the following decidability results.

Proposition 5.11. *The following decision problems can be solved effectively:*

Instance : *An NLAwtl A.*
Problem 1 : *Is the language $L(A)$ empty?*
Problem 2 : *Is the language $L(A)$ finite?*

Since universality is undecidable for linear languages (see, *e.g.*, [12]), and as one can easily design a DLAwtl for the language Σ^*, we obtain the following undecidability results.

Proposition 5.12. *The following decision problems are undecidable in general:*

(a) *Instance* : *An NLAwtl A on Σ, and a regular language R.*
Problem 3 : *Is $L(A) = \Sigma^*$, that is, is A universal?*
Problem 4 : *Is R contained in $L(A)$?*
(b) *Instance* : *Two NLAwtls A and B.*
Problem 5 : *Is $L(A)$ contained in $L(B)$?*
Problem 6 : *Is $L(A) = L(B)$?*

Finally, it has been observed in [25] that it is undecidable in general whether a given NFAwtl accepts a regular language. As the NFAwtl is a special type of NLAwtl, this immediately yields the following result.

Proposition 5.13. *The following decision problem is undecidable in general:*

Instance : *An NLAwtl A.*
Problem 7 : *Is the language $L(A)$ regular?*

Actually, we even have the following undecidability result.

Proposition 5.14. *The following decision problem is undecidable in general:*

Instance : *An NLAwtl A.*
Problem 8 : *Is the language $L(A)$ linear context-free?*

Proof. The language class $\mathcal{L}(\mathsf{NLAwtl})$ is effectively closed under union and under product with regular sets (Prop. 5.6). Further, universality is undecidable for NLAwtls. Finally, the class of linear context-free languages contains the regular languages, and it is closed under right quotients with single letters, that is, if $L \in \mathsf{LIN}$, then also $L/a = \{ w \mid wa \in L \}$ is linear context-free. Thus, we can apply Greibach's general undecidability result [11], which implies that it is undecidable in general whether the language accepted by a given NLAwtl is linear context-free. □

However, the following question remains open.

Question 2: Is it decidable whether a language that is accepted by an NLAwtl is also accepted by a DLAwtl?

Finally, observe that universality is decidable for DLAwtls, as emptiness is decidable for them by Proposition 5.11 and $\mathcal{L}(\mathsf{DLAwtl})$ is closed under complementation (Prop. 5.2). However, the following questions remain open.

Question 3: Are containment, equivalence, regularity, or linear context-freeness decidable for DLAwtls?

6. A family of languages associated to an NLA

Let $A = (Q, \Sigma, \delta, I, F)$ be an NLA in normal form. Recall from Definition 2.1 that, for each state $q \in Q$, $\mu(q) = \{ a \in \Sigma \mid \delta(q, a) \neq \emptyset \}$. Without loss of generality we may assume that

- $F = \{q_+\}$ for a state $q_+ \in Q$, that is, q_+ is a unique final state, and $\mu(q_+) = \emptyset$, that is, in this state A cannot read any letter,
- each letter $a \in \Sigma$ occurs in some word of the language $L(A)$, which means in particular that, for each $a \in \Sigma$, there exists at least one state $q \in Q$ such that $\mu(q) = \{a\}$,
- and $\mu(q) \neq \emptyset$ for each non-final state $q \in Q \setminus F$.

By introducing the sentinels $\triangleright$ and $\triangleleft$ and by adding an admissible translucency mapping $\tau : Q \to 2^\Sigma$, we obtain an NLAwtl $A_\tau = (Q, \Sigma, \triangleright, \triangleleft, \tau, \delta, I, F)$ from A. Here the translucency mapping τ is called *admissible for* A if it satisfies the following restrictions:

1. $\tau(q_+) = \emptyset$ and
2. $\tau(q) \subseteq \Sigma \setminus \mu(q)$ for all $q \in Q \setminus \{q_+\}$.

From the proof of Proposition 3.6, we see that $L(A_\tau)$ contains the linear context-free language $L = L(A)$ and that L is letter-equivalent to $L(A_\tau)$. Observe that A, and therewith the language L, does not depend in any way on the translucency mapping τ. Thus, we can choose different translucency mappings τ for the NLAwtl A_τ, but L will be contained in each of the resulting languages $L(A_\tau)$ and it will be letter-equivalent to each of these languages.

In order to investigate this phenomenon, we choose the notation (A, τ) for the NLAwtl $A_\tau = (Q, \Sigma, \triangleright, \triangleleft, \tau, \delta, I, F)$. If $|Q| = m$ and $|\Sigma| = n$, then there are $2^{(n-1)\cdot(m-1)}$ admissible translucency mappings for A, as for each of the $m - 1$ non-final states q, there are 2^{n-1} possible values for $\tau(q)$. Thus, from the given NLA A, we obtain a family

$$\mathcal{A}(A) = \{ (A, \tau) \mid \tau \text{ is an admissible translucency mapping for } A \}$$

of cardinality $2^{(n-1)\cdot(m-1)}$ of NLAwtls that are all based on the given NLA A and that only differ in their translucency mappings. Concerning the resulting family of languages we have the following basic observation, where $\tau_\emptyset$ denotes the trivial translucency mapping that is defined by taking $\tau_\emptyset(q) = \emptyset$ for all states q of A.

Proposition 6.1. *The linear context-free language $L(A) = L((A, \tau_\emptyset))$ is a sublanguage of $L((A, \tau))$ that is letter-equivalent to $L((A, \tau))$ for each NLAwtl $(A, \tau) \in \mathcal{A}(A)$.*

Let $\tau_\forall$ be the admissible translucency mapping that is defined by $\tau_\forall(q) = \Sigma \setminus \mu(q)$ for all non-final states $q \in Q \setminus F$. Then the following result is easily obtained. Just observe that each accepting computation of an NLAwtl $(A, \tau) \in \mathcal{A}(A)$ is also an accepting computation of the NLAwtl $(A, \tau_\forall)$.

Proposition 6.2. *The language $L((A, \tau_\forall))$ is the commutative closure of the language $L(A)$. In particular, $L((A, \tau))$ is a sublanguage of $L((A, \tau_\forall))$ for all $(A, \tau) \in \mathcal{A}(A)$.*

On the set of NLAwtls $\mathcal{A}(A)$, we can define a partial ordering $\leq$ as follows:

$$(A, \tau_1) \leq (A, \tau_2) \text{ iff } \forall q \in Q \setminus F : \tau_1(q) \subseteq \tau_2(q).$$

If $(A, \tau_1) \leq (A, \tau_2)$, then each accepting computation of the former is also an accepting computation of the latter. Thus, we obtain the following result.

Proposition 6.3. *If $(A, \tau_1) \leq (A, \tau_2)$, then $L((A, \tau_1)) \subseteq L((A, \tau_2))$.*

The set $\mathcal{A}(A)$ is actually a distributive lattice under the operations

$$(A, \tau_1) \vee (A, \tau_2) = (A, \tau_1 \cup \tau_2) \text{ and } (A, \tau_1) \wedge (A, \tau_2) = (A, \tau_1 \cap \tau_2),$$

where $\tau_1 \cup \tau_2$ and $\tau_1 \cap \tau_2$ are defined through

$$(\tau_1 \cup \tau_2)(q) = \tau_1(q) \cup \tau_2(q)$$

and

$$(\tau_1 \cap \tau_2)(q) = \tau_1(q) \cap \tau_2(q)$$

for all $q \in Q$.

Does this lattice of NLAwtls induce a lattice on the corresponding family of languages

$$\mathcal{L}(\mathcal{A}(A)) = \{\, L((A, \tau)) \mid (A, \tau) \in \mathcal{A}(A) \,\}?$$

Let $\Lambda : \mathcal{A}(A) \to \mathcal{L}(\mathcal{A}(A))$ be the mapping that maps each NLAwtl $(A, \tau) \in \mathcal{A}(A)$ to the corresponding language $L((A, \tau))$. We define the following binary operations $\vee$ and $\wedge$ on $\mathcal{L}(\mathcal{A}(A))$:

$$L((A, \tau_1)) \vee L((A, \tau_2)) := L((A, \tau_1) \vee (A, \tau_2))$$

and

$$L((A, \tau_1)) \wedge L((A, \tau_2)) := L((A, \tau_1) \wedge (A, \tau_2)).$$

Obviously,

$$L((A, \tau_1)) \cup L((A, \tau_2)) \subseteq L((A, \tau_1)) \vee L((A, \tau_2)),$$

as each accepting computation of (A, τ_1) or (A, τ_2) is also an accepting computation of $(A, \tau_1) \vee (A, \tau_2)$, and

$$L((A, \tau_1)) \wedge L((A, \tau_2)) \subseteq L((A, \tau_1)) \cap L((A, \tau_2)),$$

as each accepting computation of $(A, \tau_1) \wedge (A, \tau_2)$ is an accepting computation of (A, τ_1) as well as of (A, τ_2). However, in general these are proper inclusions.

Example 6.4. Let $A = (Q, \Sigma, \delta, I, F)$ be the NLA that is defined by $Q = Q_L = \{q_a, q_b, q_c, q_+\}$, $\Sigma = \{a, b, c\}$, $I = \{q_a\}$, $F = \{q_+\}$, and $\delta(q_a, a) = \{q_a, q_b\}$, $\delta(q_b, b) = \{q_b, q_c\}$, $\delta(q_c, c) = \{q_c, q_+\}$, and $\delta(q, x) = \emptyset$ for all other pairs $(q, x) \in Q \times \Sigma$. Then it is easily seen that $L(A) = a^+ \cdot b^+ \cdot c^+$.

Let τ_1 be the translucency mapping defined by $\tau_1(q_a) = \{b\}$ and $\tau_1(q_b) = \tau_1(q_c) = \emptyset$, and let τ_2 be the translucency mapping that is defined by $\tau_2(q_a) = \tau_2(q_c) = \emptyset$ and $\tau_2(q_b) = \{c\}$. Then

$$L((A, \tau_1)) = \{\, w \in \{a, b\}^+ \mid |w|_a, |w|_b \geq 1 \,\} \cdot c^+,$$

and

$$L((A, \tau_2)) = a^+ \cdot \{\, w \in \{b, c\}^+ \mid |w|_b, |w|_c \geq 1 \,\}.$$

Now let $\tau_3 = \tau_1 \cup \tau_2$. Then $\tau_3(q_a) = \{b\}$, $\tau_3(q_b) = \{c\}$, and $\tau_3(q_c) = \emptyset$, and $w = bacb \in L((A, \tau_3)) \smallsetminus (L((A, \tau_1)) \cup L((A, \tau_2)))$, that is,

$$L((A, \tau_1)) \cup L((A, \tau_2)) \subsetneq L((A, \tau_1)) \vee L((A, \tau_2)).$$

Example 6.5. Let $A = (Q, \Sigma, \delta, I, F)$ be the NLA that is defined by $Q = Q_L = \{q_a, q'_a, q_b, q_+\}$, $\Sigma = \{a, b\}$, $I = \{q_a, q'_a\}$, $F = \{q_+\}$, and $\delta(q_a, a) = \{q_a, q_b\}$, $\delta(q'_a, a) = \{q'_a, q_b\}$, $\delta(q_b, b) = \{q_b, q_+\}$, and $\delta(q, x) = \emptyset$ for all other pairs $(q, x) \in Q \times \Sigma$. Then it is easily seen that $L(A) = a^+ \cdot b^+$.

Let τ_1 be the translucency mapping defined by $\tau_1(q_a) = \{b\}$ and $\tau_1(q'_a) = \tau_1(q_b) = \emptyset$, and let τ_2 be the translucency mapping that is defined by $\tau_2(q_a) = \tau_2(q_b) = \emptyset$ and $\tau_2(q'_a) = \{b\}$. Then

$$L((A, \tau_1)) = \{\, w \in \{a, b\}^+ \mid |w|_a, |w|_b \geq 1 \,\} = L((A, \tau_2)).$$

Now let $\tau_3 = \tau_1 \cap \tau_2$. Then $\tau_3(q_a) = \tau_3(q'_a) = \tau_3(q_b) = \emptyset$, and $L((A, \tau_3)) = L(A) = a^+ \cdot b^+ \subsetneq L((A, \tau_1)) \cap L((A, \tau_2)) = L((A, \tau_1))$, that is,

$$L((A, \tau_1)) \wedge L((A, \tau_2)) \subsetneq L((A, \tau_1)) \cap L((A, \tau_2)).$$

The next examples illustrate the possible structures of the set $\mathcal{L}(\mathcal{A}(A))$.

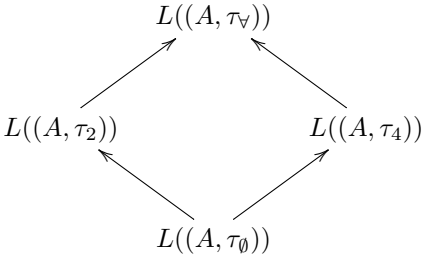
Example 6.6. Let $A = (Q, \Sigma, \delta, I, F)$ be the NLA that is defined by $Q = Q_L = \{q_a, q_b, q_+\}$, $\Sigma = \{a, b\}$, $I = \{q_a, q_b\}$, $F = \{q_+\}$, and $\delta(q_a, a) = Q$, $\delta(q_b, b) = Q$, and $\delta(q, x) = \emptyset$ for all other pairs $(q, x) \in Q \times \Sigma$. Then it is easily seen that $L(A) = \{a, b\}^+$. Further, $L(A, \tau_\forall) = \{a, b\}^+$ as well, which implies that $L(A, \tau) = \{a, b\}^+$ for all admissible translucency mappings τ. Thus, the language class $\mathcal{L}(\mathcal{A}(A))$ consists of only one language, the language $\{a, b\}^+$.

Example 6.7. Let $A = (Q, \Sigma, \delta, I, F)$ be the NLA that is defined by $Q = Q_L = \{q_a, q_b, q_c, q_+\}$, $\Sigma = \{a, b\}$, $I = \{q_a\}$, $F = \{q_+\}$, and $\delta(q_a, a) = \{q_a, q_b\}$, $\delta(q_b, b) = \{q_b, q_c\}$, $\delta(q_c, a) = \{q_c, q_+\}$, and $\delta(q, x) = \emptyset$ for all other pairs $(q, x) \in Q \times \Sigma$. Then it is easily seen that $L(A) = a^+ \cdot b^+ \cdot a^+$.

As we have three non-final states and two input letters, there are 2^8 possible translucency mappings for A, which are listed in the following table together with the resulting languages:

τ	q_a	q_b	q_c	$L((A,\tau))$
$\tau_\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$a^+ \cdot b^+ \cdot a^+$
τ_1	$\emptyset$	$\emptyset$	$\{b\}$	$a^+ \cdot b^+ \cdot a^+$
τ_2	$\emptyset$	$\{a\}$	$\emptyset$	$a \cdot \{\, w \in \{a,b\}^+ \mid \lvert w\rvert_a, \lvert w\rvert_b \geq 1 \,\}$
τ_3	$\emptyset$	$\{a\}$	$\{b\}$	$a \cdot \{\, w \in \{a,b\}^+ \mid \lvert w\rvert_a, \lvert w\rvert_b \geq 1 \,\}$
τ_4	$\{b\}$	$\emptyset$	$\emptyset$	$\{\, w \in \{a,b\}^+ \mid \lvert w\rvert_a, \lvert w\rvert_b \geq 1 \,\} \cdot a$
τ_5	$\{b\}$	$\emptyset$	$\{b\}$	$\{\, w \in \{a,b\}^+ \mid \lvert w\rvert_a, \lvert w\rvert_b \geq 1 \,\} \cdot a$
τ_6	$\{b\}$	$\{a\}$	$\emptyset$	$\{\, w \in \{a,b\}^+ \mid \lvert w\rvert_a \geq 2, \lvert w\rvert_b \geq 1 \,\}$
$\tau_\forall$	$\{b\}$	$\{a\}$	$\{b\}$	$\{\, w \in \{a,b\}^+ \mid \lvert w\rvert_a \geq 2, \lvert w\rvert_b \geq 1 \,\}$

Here we see that $L((A,\tau_\emptyset)) = L((A,\tau_1))$, $L((A,\tau_2)) = L((A,\tau_3))$, $L((A,\tau_4)) = L((A,\tau_5))$, and $L((A,\tau_6)) = L((A,\tau_\forall))$, and that with respect to inclusion these families are related as shown in the following diagram:



Observe that in Example 6.7, τ_1 and τ_2 are incomparable with respect to inclusion, but that $L((A,\tau_1)) \subsetneq L((A,\tau_2))$ holds, and the same is true for τ_5 and τ_6. So the following question remains.

Question 4: Under what conditions is the set $\mathcal{L}(\mathcal{A}(A))$ a distributive lattice under the operations $\vee$ and $\wedge$ defined above?

Based on an NLA in normal form, we have defined a class of NLAwtls and a corresponding class of languages by adjoining all admissible translucency mappings. Of course, the same definition can be applied to NFAs to define a class of NFAwtls and the corresponding class of languages. Observe that the NLAs in Examples 6.4–6.7 above only contain left states, and hence, they are actually NFAs. Thus, Question 4 extends to NFAs.

7. Conclusion

We have extended the nondeterministic linear automaton by translucency mappings, and we have seen that the resulting NLAwtls are quite expressive. In fact, they accept a subclass of the class of all languages with semi-linear Parikh image that properly contains all linear context-free trace languages. In addition, we successfully characterized the class of all linear context-free trace languages by a restricted type of NLAwtls. Further, we presented a number of closure and non-closure results for the language classes $\mathcal{L}(\mathsf{NLAwtl})$ and $\mathcal{L}(\mathsf{DLAwtl})$, but some operations still remain to be considered. Also we considered various decision problems for NLAwtls. Finally, we obtained a finite lattice of NLAwtls $\mathcal{A}(A)$ from each NLA A that is in normal form, and this lattice yields a finite family of languages $\mathcal{L}(\mathcal{A}(A))$ that are partially ordered with respect to inclusion. However, it remains to study these families of languages in more detail.

References

[1] I. Aalbersberg and G. Rozenberg. Theory of traces. *Theor. Comput. Sci.* **60** (1988) 1–82.
[2] J.M. Autebert, J. Berstel and L. Boasson. Context-free languages and pushdown automata, in Vol. 1 of Handbook of Formal Languages, edited by G. Rozenberg and A. Salomaa. Springer, Berlin (1997) 111–174.
[3] J. Berstel. Transductions and Context-Free Languages. Leitfäden der angewandten Mathematik und Mechanik, Vol. 38. Teubner, Stuttgart (1979).

[4] A. Bertoni, G. Mauri and N. Sabadini, Membership problems for regular and context-free trace languages. *Inf. Comput.* **82** (1989) 135–150.

[5] G. Buntrock and F. Otto, Growing context-sensitive languages and Church-Rosser languages. *Inf. Comput.* **141** (1998) 1–36.

[6] E. Czeizler and E. Czeizler, A short survey on Watson-Crick automata. *Bull. EATCS* **88** (2006) 104–119.

[7] E. Dahlhaus and M. Warmuth, Membership for growing context-sensitive grammars is polynomial. *J. Comput. Syst. Sci.* **33** (1986) 456–472.

[8] V. Diekert and G. Rozenberg, The Book of Traces. World Scientific, Singapore (1995).

[9] H. Fernau, M. Paramasivan and M.L. Schmid, Jumping finite automata: Characterizations and complexity, *CIAA 2012, Proc.*, edited by F. Drewes. In Vol. 9223 of *Lect. Notes Comput. Sci.* Springer, Heidelberg (2012) 89–101.

[10] R. Freund, Gh. Păun, G. Rozenberg and A. Salomaa, Watson-Crick finite automata, DNA Based Computers, *Proc.*, *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, edited by H. Rubin and D.H. Wood. DIMACS/AMS, Rhode Island, USA (1999) 297–328.

[11] S. Greibach, A note on undecidable properties of formal languages. *Math. Syst. Theory* **2** (1968) 1–6.

[12] J.E. Hopcroft and J.D. Ullman, Introduction to Automata Theory, Languages, and Computation. Addison-Wesley, Reading, MA (1979).

[13] R. Kocman, B. Nagy, Z. Krivka and A. Meduna, A jumping $5'\to3'$ Watson-Crick finite automata model, *Tenth Workshop on Non-Classical Models of Automata and Applications (NCMA 2018), Proc.*, edited by R. Freund, M. Hospodár, G. Jirásková and G. Pighizzini. Österreichische Computer Gesellschaft (2018) 117–132.

[14] P. Leupold and B. Nagy, $5' \to 3'$ Watson-Crick automata with several runs. *Fund. Inf.* **104** (2010) 71–91.

[15] R. Loukanova, Linear context free languages, *ICTAC 2007, Proc.*, edited by C.B. Jones, Z. Liu, and J. Woodcock. In Vol. 4711 of *Lect. Notes Comput. Sci.* Springer, Heidelberg (2007) 351–365.

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

[17] B. Nagy, On $5' \to 3'$ sensing Watson-Crick automata, DNA Computing, *13th International Meeting (2007),* Revised Selected Papers, edited by M. Garzon and H. Yan. In Vol. 4848 of *Lect. Notes Comput. Sci.* Springer, Heidelberg (2008) 256–262.

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

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

[20] B. Nagy and L. Kovács, Finite automata with translucent letters applied in natural and formal language theory, Transactions on Computational Collective Intelligence XVII, edited by N.T. Nguyen, R. Kowalczyk, A. Fred and F. Joaquim. In Vol. 8790 of *Lect. Notes Comput. Sci.* Springer, Heidelberg (2014), pages 107–127.

[21] B. Nagy and Z. Kovács, On simple $5'\to3'$ sensing Watson-Crick finite-state transducers, *Eleventh Workshop on Non-Classical Models of Automata and Applications (NCMA 2019), Proc.*, edited by R. Freund, M. Holzer and J.M. Sempere. Österreichische Computer Gesellschaft (2019) 155–170.

[22] B. Nagy and F. Otto, CD-systems of stateless deterministic R(1)-automata accept all rational trace languages. *LATA 2010, Proc.*, edited by A.H. Dediu, H. Fernau and C. Martin-Vide. In Vol. 6031 of *Lect. Notes Comput. Sci.* Springer, Heidelberg (2010) 463–474.

[23] B. Nagy and F. Otto, An automata-theoretical characterization of context-free trace languages, *SOFSEM 2011, Proc.*, edited by I. Cerná, T. Gyimóthy, J. Hromkovic, K.G. Jeffery, R. Královic, M. Vukolic and S. Wolf. In Vol. 6543 of *Lect. Notes Comput. Sci.* Springer, Heidelberg (2011) 406–417.

[24] B. Nagy and F. Otto, CD-systems of stateless deterministic R(1)-automata governed by an external pushdown store. *RAIRO: ITA* **45** (2011) 413–448.

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

[26] 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.

[27] 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.

[28] B. Nagy and F. Otto, Two-head finite-state acceptors with translucent letters, *SOFSEM 2019, Proc.*, edited by B. Catania, R. Královič, J. Nawrocki and G. Pighizzini. In Vol. 11376 of *Lect. Notes Comput. Sci.* Springer, Heidelberg (2019) 406–418.

[29] B. Nagy and S. Parchami, On deterministic sensing $5' \to 3'$ Watson-Crick finite automata: A full hierarchy in 2detlin. *Acta Inf.* DOI: 10.1007/s00236-019-00362-6 (2020).

[30] B. Nagy, S. Parchami and H.M.M. Sadeghi, A new sensing $5' \to 3'$ Watson-Crick automata concept, *15th International Conference on Automata and Languages (AFL 2017)*, in Vol. 252 of *EPTCS*, edited by E. Csuhaj-Varjú, P. Dömösi and Gy. Vaszil (2017) 195–204.

[31] S. Parchami and B. Nagy, Deterministic sensing $5' \to 3'$ Watson-Crick automata without sensing parameter, Unconventional Computation and Natural Computation - 17th Int. Conf. (UCNC 2018) Proc., edited by S. Stepney and S. Verlan. In Vol. 10867 of *Lect. Notes Comput. Sci.* Springer, Heidelberg (2018) 173–187.

[32] Gh. Păun, G. Rozenberg and A. Salomaa, DNA Computing - New Computing Paradigms. Springer, Heidelberg (1998).

[33] A.L. Rosenberg, On multi-head finite automata. *IBM J. Res. Dev.* **10** (1966) 388–394.

[34] G. Rozenberg and A. Salomaa, Handbook of Formal Languages. Springer, Heidelberg (1997).

[35] M. Sipser, Introduction to the Theory of Computation. PWS Publishing Company, Boston, MA (1997).