Finite-state Acceptors with Translucent Letters*

Benedek Nagy[1] and Friedrich Otto[2]

[1]Department of Computer Science, Faculty of Informatics, University of Debrecen
4032 Debrecen, Egyetem tér 1., Hungary

[2]Fachbereich Elektrotechnik/Informatik, Universität Kassel, 34109 Kassel, Germany

Abstract. Finite-state acceptors with translucent letters are presented. These devices do not read their input strictly from left to right as in the traditional setting, but for each internal state of such a device, certain letters are translucent, that is, in this state the acceptor cannot see them. We describe the computational power of these acceptors, both in the deterministic and in the nondeterministic case. The languages accepted have semi-linear Parikh images, and the nondeterministic acceptors are sufficiently expressive to accept all rational trace languages. However, in contrast to the classical finite-state acceptors, the deterministic acceptors are less expressive than the nondeterministic ones.

1 Introduction

The finite-state acceptor is a fundamental computing device for accepting languages. Its deterministic version (DFA) and its nondeterministic version (NFA) both accept exactly the regular languages, and they are being used in many areas like compiler construction, text editors, hardware design, etc. Of course, these acceptors are much too weak for many applications, as the expressiveness of regular languages is quite limited. Accordingly, much more powerful models of automata have been introduced and studied like, e.g., pushdown automata, linear-bounded automata, and Turing acceptors (machines). But this larger expressive power comes at a price in that certain algorithmic questions like the word problem or the emptiness problem become more complex or even undecidable. Hence, when dealing with applications, for example in natural language processing or concurrency control, it is of importance to find models of automata that reconcile two contrasting goals: sufficient expressiveness and a moderate degree of complexity.

In the field of natural language processing this has led to the formulation of the notion of "mildly context-sensitive languages" [5, 6]. These form a subclass of the context-sensitive languages that is much more expressive than the context-free languages. They

* This work was supported by grants from the Balassi Intézet Magyar Ösztöndíj Bizottsága (MÖB) and the Deutsche Akademischer Austauschdienst (DAAD). The first author was also supported by the TÁMOP 4.2.1/B-09/1/KONV-2010-0007 project, which is implemented through the New Hungary Development Plan, co-financed by the European Social Fund and the European Regional Development Fund.

Nagy B. and Otto F..
Finite-state Acceptors with Translucent Letters.
DOI: 10.5220/0003272500030013
In *Proceedings of the 1st International Workshop on AI Methods for Interdisciplinary Research in Language and Biology* (BILC-2011), pages 3-13
ISBN: 978-989-8425-42-3

contain some typical examples of non-context-free languages like $\{ a^n b^n c^n \mid n \geq 0 \}$ and $\{ a^n b^m c^n d^m \mid m, n \geq 0 \}$, but at the same time they share many of the nice properties with the context-free languages. For example, they have semi-linear Parikh images and their parsing complexity is polynomially bounded.

In the study of concurrent systems, the theory of *traces* is an important area (see, e.g., [2]), which has also received much attention in formal language theory. For trace monoids the notions of recognizable and rational subsets do not coincide in general. For recognizable trace languages a characterization in terms of automata is given through Zielonka's asynchronous automata (see [2]), while for rational trace languages a characterization in terms of automata has only been given recently in terms of a special type of cooperating distributed systems of restarting automata, the so-called *CD-systems of stateless deterministic R(1)-automata* [7].

The restarting automaton is a formal device motivated by considerations from linguistics (see, e.g., [4]), while CD-systems of formal systems are closely related to distributed computing and multiagent systems. In fact, CD-systems of stateless deterministic R(1)-automata are related to the *binding-blocking automata* of [1], which are biologically motivated devices that do not process their input strictly in left-to-right order.

Here we describe an extension of the finite-state acceptor that vastly increases its expressive power, the so-called "finite-state acceptor with translucent letters" (*NFAwtl* for short). An NFAwtl does not read its input strictly from left to right as in the traditional setting, but for each of its internal states, certain letters are translucent, that is, in this state the NFAwtl cannot see them. Accordingly, it may read (and erase) a letter from the middle or the end of the given input. Here we consider the computational power of these acceptors. They accept certain non-regular and even some non-context-free languages, but all languages accepted by NFAwtls have semi-linear Parikh images. In fact, NFAwtls are sufficiently expressive to accept all rational trace languages, but they also accept some languages that are not rational trace languages. In contrast to the classical finite-state acceptors, the deterministic variants of the NFAwtls, the so-called *DFAwtls*, are less expressive than the nondeterministic ones, as DFAwtls do not accept all rational trace languages. Finally, closure and non-closure properties for the class of languages accepted by NFAwtls and some decidability and undecidability results for them are presented.

Most of these results follow from translations of our acceptors with translucent letters to the CD-systems of stateless deterministic R(1)-automata and back. However, we think that our acceptors are much simpler than these CD-systems, and that they are more easily accessible to non-experts.

This paper is structured as follows. In Section 2 we introduce the finite-state acceptor with translucent letters, we present two examples, and we state the first results on their expressive power. In Section 3 we consider rational trace languages, and in Section 4 we present closure and non-closure results as well as some decidability and undecidability results. In Section 5 we summarize our work and point to a number of open problems for future work. Finally, in Section 6 we explain in detail how our finite-state acceptors with translucent letters relate to the CD-systems of stateless deterministic R(1)-automata of [7].

2 Finite-state Acceptors with Translucent Letters

Here we introduce the announced variant of the nondeterministic finite-state acceptor.

Definition 1. *A* finite-state acceptor with translucent letters *(NFAwtl) is defined as a 7-tuple* $A = (Q, \Sigma, \$, \tau, I, F, \delta)$*, where* Q *is a finite set of internal states,* Σ *is a finite alphabet of input letters,* $\$ \notin \Sigma$ *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 *does not see these letters.* A *is called* deterministic*, abbreviated as DFAwtl, if* $|I| = 1$ *and if* $|\delta(q, a)| \leq 1$ *for all* $q \in Q$ *and all* $a \in \Sigma$*.*

An NFAwtl $A = (Q, \Sigma, \$, \tau, I, F, \delta)$ 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 $w \cdot \$$ on its input tape. Assume that $w = a_1 a_2 \cdots a_n$ for some $n \geq 1$ and $a_1, \ldots, a_n \in \Sigma$. Then A looks for the first occurrence from the left of a letter that is not translucent for state q_0, that is, 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 $uv \cdot \$$, 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 \$-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 $Q \cdot \Sigma^* \cdot \$$ of configurations:

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

Observe that this definition also applies to configurations of the form $q \cdot \$$, that is, $q \cdot \varepsilon \cdot \$ \vdash_A \mathsf{Accept}$ 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 $q_0 w \cdot \$ \vdash_A^* \mathsf{Accept}$, 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, $\mathcal{L}(\mathsf{NFAwtl})$ denotes the class of all languages that are accepted by NFAwtls, and $\mathcal{L}(\mathsf{DFAwtl})$ denotes the class of all languages that are accepted by DFAwtls.

The classical *nondeterministic finite-state acceptor* (NFA) is obtained from the NFAwtl by removing the endmarker \$ and by ignoring the translucency relation τ, and the *deterministic finite-state acceptor* (DFA) is obtained from the DFAwtl in the same way. Thus, the NFA (DFA) can be interpreted as a special type of NFAwtl (DFAwtl). Accordingly, we have the following, where REG denotes the class of regular languages.

Lemma 1. $\mathsf{REG} \subseteq \mathcal{L}(\mathsf{DFAwtl}) \subseteq \mathcal{L}(\mathsf{NFAwtl})$.

However, already DFAwtls are much more expressive than standard DFAs.

Example 1. Let $A = (Q, \Sigma, \$, \tau, I, F, \delta)$, where $Q = \{q_0, q_1, q_2\}$, $I = \{q_0\} = F$, $\Sigma = \{a, b, c\}$, and the functions τ and δ are defined as follows:

$$\begin{array}{ll} \tau(q_0) = \emptyset, & \delta(q_0, a) = \{q_1\}, \\ \tau(q_1) = \{a, c\}, & \delta(q_1, b) = \{q_2\}, \\ \tau(q_2) = \{a, b\}, & \delta(q_2, c) = \{q_0\}, \end{array}$$

and $\delta(q, x) = \emptyset$ for all other pairs $(q, x) \in Q \times \Sigma$. Observe that A is in fact a DFAwtl.

Given the word $w = abaccb$ as input, A will execute the following computation:

$$\begin{array}{c} q_0abaccb \cdot \$ \vdash_A q_1baccb \cdot \$ \vdash_A q_2accb \cdot \$ \vdash_A q_0acb \cdot \$ \vdash_A q_1cb \cdot \$ \\ \vdash_A q_2c \cdot \$ \quad \vdash_A q_0 \cdot \$ \quad \vdash_A \mathsf{Accept}, \end{array}$$

as q_0 is a final state. In fact, it can be shown that

$$\begin{array}{l} L(A) = \{\, w \in \Sigma^* \mid |w|_a = |w|_b = |w|_c \geq 0, \\ \qquad\qquad \text{and for each prefix } u \text{ of } w : |u|_a \geq \max\{|u|_b, |u|_c\} \,\}. \end{array}$$

This language is not context-free, as $L(A) \cap (a^* \cdot b^* \cdot c^*) = \{\, a^n b^n c^n \mid n \geq 0 \,\}$.

Thus, already DFAwtls accept non-context-free languages, that is, we have the proper inclusion $\mathsf{REG} \subsetneq \mathcal{L}(\mathsf{DFAwtl})$.

An NFAwtl $A = (Q, \Sigma, \$, \tau, I, F, \delta)$ can be described by a graph, similar to the graph representation of NFAs. A state $q \in Q$ is represented by a node labelled q, where the node of an initial state p is marked by a special incoming edge without a label, and the node of a final state p is marked by a special outgoing edge with label $(\tau(p))^*$. For each state $q \in Q$ and each letter $a \in \Sigma \setminus \tau(q)$, if $\delta(q, a) = \{q_1, \ldots, q_s\}$, then there is a directed edge labelled $((\tau(q))^*, a)$ from the node corresponding to state q to the node corresponding to state q_i for each $i = 1, \ldots, s$. The graph representation of the DFAwtl A of Example 1 is given in Figure 1, where $\{\varepsilon\}$ is used instead of $\emptyset^*$.

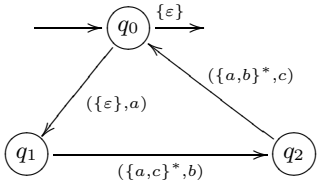

Fig. 1. The graphical representation of the DFAwtl A of Example 1.

Example 2. An NFTwtl for the language

$$\{\, a^n w \mid n \geq 1,\ w \in \{a, b, c\}^+ \text{ satisfying } |w|_a = |w|_b = |w|_c \,\}$$

is described by the diagram in Figure 2.

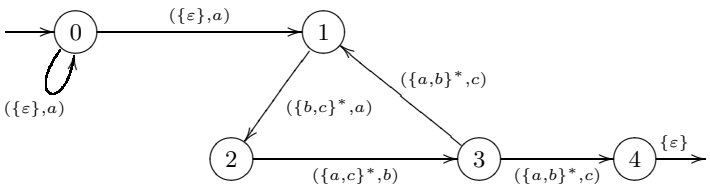

Fig. 2. The graphical representation of an NFAwtl accepting the language $\{\, a^n w \mid n \geq 1,\ w \in \{a,b,c\}^+ \text{ satisfying } |w|_a = |w|_b = |w|_c \,\}$.

According to the definition above, an NFAwtl may accept a word without processing it completely. For example, the NFAwtl $A_1 = (\{q\}, \{a\}, \$, \tau, \{q\}, \{q\}, \delta)$ defined by $\tau(q) = \{a\}$ and $\delta(q, a) = \emptyset$ accepts each word from a^* in a single step. This, however, is only a convenience, as shown by the following normal form result.

Proposition 1. *From a given NFAwtl $A = (Q, \Sigma, \$, \tau, I, F, \delta)$ one can effectively construct an NFAwtl $B = (Q', \Sigma, \$, \tau', I', F', \delta')$ such that $L(B) = L(A)$, but for each word $w \in L(B)$, each accepting computation of B on input w consists of $|w|$ many reading steps plus a final step that accepts the empty word.*

Observe that the DFAwtl of Example 1 and the NFAwtl of Example 2 are in normal form. Nevertheless it remains open whether Proposition 1 holds for DFAwtls in general. If A is an NFAwtl on Σ that is in normal form, then by removing the translucency relation from A, we obtain a standard NFA A' that has the following properties.

Proposition 2. *By removing the translucency relation from an NFAwtl A in normal form, we obtain an NFA A' such that $L(A')$ is a subset of $L(A)$ that is letter-equivalent to $L(A)$.*

Here two languages over the same alphabet $\Sigma = \{a_1, \ldots, a_n\}$ 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 $\mathcal{L}(\mathsf{NFAwtl})$ is letter-equivalent to a regular language. This yields the following consequence.

Corollary 1. *All languages in $\mathcal{L}(\mathsf{NFAwtl})$ are semi-linear, that is, if $L \subseteq \Sigma^*$ is accepted by some NFAwtl, then $\psi(L)$ is a semi-linear subset of $\mathbb{N}^n$.*

As the languages $\{\, a^n b^n \mid n \geq 0 \,\}$ and $\{\, a^n b^n c^n \mid n \geq 0 \,\}$ do not contain regular sublanguages that are letter-equivalent to the languages themselves, it follows from Proposition 2 that these languages are not accepted by any NFAwtls.

3 Rational 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., [2]).

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., [2]).

If $M(D)$ is a trace monoid generated by Σ, then we use φ_D to denote the morphism $\varphi_D : \Sigma^* \to M(D)$ that is defined by $w \mapsto [w]_D$ for all words $w \in \Sigma^*$. 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), that is, 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 $\mathcal{LRAT}$ we denote the class of all rational trace languages.

Theorem 1. $\mathcal{LRAT} \subsetneq \mathcal{L}(\mathsf{NFAwtl})$.

The above inclusion is proper as the Dyck language $D_1'^*$, which is not a rational trace language, is accepted by a DFAwtl. On the other hand, the following technical result shows that Theorem 1 does not extend to DFAwtls.

Proposition 3. *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.

It follows that $\mathcal{L}(\mathsf{DFAwtl})$ and $\mathcal{LRAT}$ are incomparable under inclusion.

4 Closure Properties and Decidability Results

Proposition 4.

(a) *The language class $\mathcal{L}(\mathsf{NFAwtl})$ is closed under union, product, and Kleene star.*
(b) *The language class $\mathcal{L}(\mathsf{NFAwtl})$ is neither closed under intersection with regular languages, nor under complementation, nor under ε-free morphisms.*

The *commutative closure* $\mathrm{com}(L)$ of a language $L \subseteq \Sigma^*$ is the set of all words that are letter-equivalent to a word from L, that is,

$$\mathrm{com}(L) = \psi^{-1}(\psi(L)) = \{ w \in \Sigma^* \mid \exists\, u \in L : \psi(w) = \psi(u) \},$$

where $\psi : \Sigma^* \to \mathbb{N}^{|\Sigma|}$ denotes the Parikh mapping. If L is accepted by an NFAwtl A in normal form, then from A we obtain a finite-state acceptor A' for a regular sublanguage E of L that is letter-equivalent to L by ignoring the transparency relation (Proposition 2). Obviously, the commutative closure $\mathrm{com}(L)$ of L coincides with the commutative closure $\mathrm{com}(E)$ of E. For the dependency relation $D = \{ (a, a) \mid a \in \Sigma \}$, the trace monoid $M(D)$ is the free commutative monoid generated by Σ. Thus, $\mathrm{com}(E) = \bigcup_{w \in E} [w]_D$ is simply the rational trace language $\varphi_D^{-1}(\varphi_D(E))$. Hence, it follows from Theorem 1 that this language is accepted by an NFAwtl B.

Corollary 2. *The language class $\mathcal{L}(\mathsf{NFAwtl})$ is closed under the operation of taking the commutative closure.*

A language $L \subseteq \Sigma^*$ is called *commutative* if $\mathrm{com}(L) = L$ holds, that is, if it contains all permutations of all its elements. As each semi-linear language is letter-equivalent to some regular language, it follows that each commutative semi-linear language is the commutative closure of some regular language, and therewith it is a rational trace language. Thus, Theorem 1 implies the following result.

Corollary 3. *All commutative semi-linear languages are contained in $\mathcal{L}(\mathsf{NFAwtl})$.*

If $L_1 \subseteq \Sigma^*$ and $L_2 \subseteq \Gamma^*$ are languages, then the *shuffle* of L_1 and L_2 is defined as

$$\mathrm{sh}(L_1, L_2) = \{ u_1 v_1 u_2 \cdots u_n v_n \mid n \geq 1, u_1 \cdots u_n \in L_1, v_1 \cdots v_n \in L_2 \}.$$

If Σ and Γ are disjoint, then $\mathrm{sh}(L_1, L_2)$ is called a *disjoint shuffle*.

Proposition 5. *The language class $\mathcal{L}(\mathsf{NFAwtl})$ is closed under disjoint shuffle.*

However, it is still open whether the language class $\mathcal{L}(\mathsf{NFAwtl})$ is closed under inverse morphisms or under the operation of reversal. An NFAwtl can easily be simulated by a nondeterministic one-tape Turing machine that is simultaneously linearly space-bounded and quadratically time-bounded. Hence, we have the complexity result.

Proposition 6. $\mathcal{L}(\mathsf{NFAwtl}) \subseteq \mathsf{NSpaceTime}(n, n^2)$.

It follows in particular that $\mathcal{L}(\mathsf{NFAwtl})$ only contains context-sensitive languages. Proposition 2 yields an effective construction of a finite-state acceptor A' from an NFAwtl 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 finite-state acceptors, this immediately yields the following decidability results.

Proposition 7. *The following decision problems are effectively decidable:*
Instance : *An NFAwtl A.*
Question 1 : *Is the language $L(A)$ empty?*
Question 2 : *Is the language $L(A)$ finite?*

On the other hand, it is undecidable in general whether a rational trace language is recognizable (see, e.g., [2]). As a rational subset S of a trace monoid $M(D)$ is recognizable if and only if $\varphi_D^{-1}(S)$ is a regular language, we obtain the following undecidability result from Theorem 1.

Proposition 8. *The following decision problem is undecidable in general:*
Instance : *An NFAwtl A.*
Question : *Is the language $L(A)$ regular?*

By a reduction from the universality problem for rational transducers, which is undecidable in general [3], also the following undecidability results can be derived.

Proposition 9. *The following problems are undecidable in general:*
Instance : *Two NFAwtls A_1 and A_2.*
Question 1 : *Is $L(A_1)$ contained in $L(A_2)$?*
Question 2 : *Are A_1 and A_2 equivalent, that is, does $L(A_1) = L(A_2)$ hold?*

5 Conclusions

We have presented finite-state acceptors with translucent letters, and we have seen that they are quite expressive. In fact they accept a subclass of the class of all languages with semi-linear Parikh image that properly contains all rational trace languages. As the deterministic variants of our acceptors do not accept all rational trace languages, it remains to characterize their expressive power. In particular, which rational trace languages are accepted by DFAwtls? Then we have given a number of closure and non-closure results for the language class $\mathcal{L}(\mathsf{NFAwtl})$, but it remains open whether or not this class is closed under inverse morphisms or under the operation of reversal. Also the closure and non-closure properties of the class $\mathcal{L}(\mathsf{DFAwtl})$ remain to be studied.

It appears that the fact that the translucency mapping only depends on the actual state but not on the letter actually processed is one of the reasons for the limited expressive power of DFAwtls. Hence, we plan to also study a variant of the DFAwtl in which the translucency mapping depends on the actual state and the letter read (erased).

6 Proofs

The NFAwtl considered here is closely related to the so-called *cooperating distributed system of stateless deterministic R*(1)*-automata* that was introduced in [7].

A *stateless deterministic R*(1)*-automaton* is a one-tape machine with a read/write window of size 1. Formally it is described by a 5-tuple $M = (\Sigma, ¢, \$, 1, \delta)$, where Σ is a finite alphabet, the symbols $¢, \$ \notin \Sigma$ serve as markers for the left and right border of the work space, respectively, and $\delta : \Sigma \cup \{¢, \$\} \to \{\mathsf{MVR}, \mathsf{Accept}, \varepsilon, \emptyset\}$ is a (partial) *transition function*. There are three types of transition steps: *move-right steps* (MVR), which shift the window one step to the right, combined *rewrite/restart steps* (ε), which delete the content a ($a \in \Sigma$) of the window, thereby shortening the tape, and place

the window over the left end of the tape, and *accept steps* (Accept), which cause the automaton to halt and accept. If $\delta(a) = \emptyset$, then no step is possible and the automaton halts and rejects.

A *configuration* of M is described by a pair (α, β), where either $\alpha = \varepsilon$ and $\beta \in \{\text{¢}\} \cdot \Sigma^* \cdot \{\$\}$ or $\alpha \in \{\text{¢}\} \cdot \Sigma^*$ and $\beta \in \Sigma^* \cdot \{\$\}$; here $\alpha\beta$ is the current content of the tape, and it is understood that the head scans the first symbol of β. A *restarting configuration* is of the form $(\varepsilon, \text{¢}w\$)$, where $w \in \Sigma^*$. By $\vdash_M$ we denote the single-step computation relation of M, and $\vdash_M^*$ denotes the reflexive transitive closure of $\vdash_M$.

The automaton M proceeds as follows. Starting from an initial configuration $(\varepsilon, \text{¢}w\$)$, the window moves right until a configuration of the form $(\text{¢}x, ay\$)$ is reached such that $\delta(a) = \varepsilon$. Now the latter configuration is transformed into the restarting configuration $(\varepsilon, \text{¢}xy\$)$. This computation, which is called a *cycle*, is expressed as $w \vdash_M^c xy$. A computation of M now consists of a finite sequence of cycles that is followed by a tail computation, which consists of a sequence of move-right operations possibly followed by an accept step. An input word $w \in \Sigma^*$ is *accepted* by M, if the computation of M which starts with the initial configuration $(\varepsilon, \text{¢}w\$)$ finishes by executing an accept step. By $L(M)$ we denote the language consisting of all words accepted by M.

If $M = (\Sigma, \text{¢}, \$, 1, \delta)$ is a stateless deterministic R(1)-automaton, then we can partition its alphabet Σ into four disjoint subalphabets:

$$(1.)\ \Sigma_1 = \{\, a \in \Sigma \mid \delta(a) = \mathsf{MVR} \,\}, \quad (3.)\ \Sigma_3 = \{\, a \in \Sigma \mid \delta(a) = \mathsf{Accept} \,\},$$
$$(2.)\ \Sigma_2 = \{\, a \in \Sigma \mid \delta(a) = \varepsilon \,\}, \quad (4.)\ \Sigma_4 = \{\, a \in \Sigma \mid \delta(a) = \emptyset \,\}.$$

A *cooperating distributed system of stateless deterministic R*(1)*-automata* (or a stl-det-local-CD-R(1)-system for short) consists of a finite collection $\mathcal{M} = ((M_j, \sigma_j)_{j \in J}, J_0)$ of stateless deterministic R(1)-automata $M_j = (\Sigma, \text{¢}, \$, 1, \delta_j)$ ($j \in J$), *successor relations* $\sigma_j \subseteq J$ ($j \in J$), and a subset $J_0 \subseteq J$ of *initial indices*. Here it is required that $J_0 \neq \emptyset$, and that $\sigma_j \neq \emptyset$ for all $j \in J$. Various modes of operation have been introduced and studied, but here we are only interested in mode $= 1$ computations.

A computation of $\mathcal{M}$ (in mode $= 1$) on an input word w proceeds as follows. First an index $j_0 \in J_0$ is chosen nondeterministically. Then the R-automaton M_{j_0} starts the computation with the initial configuration $(\varepsilon, \text{¢}w\$)$, and executes a single cycle. Thereafter an index $j_1 \in \sigma_{j_0}$ is chosen nondeterministically, and M_{j_1} continues the computation by executing a single cycle. This continues until, for some $l \geq 0$, the machine M_{j_l} accepts. Should at some stage the chosen machine M_{j_l} be unable to execute a cycle or to accept, then the computation fails. By $L(\mathcal{M})$ we denote the language that the system $\mathcal{M}$ accepts. By $\mathcal{L}$(stl-det-local-CD-R(1)) we denote the class of languages that are accepted by stl-det-local-CD-R(1)-systems.

If $A = (Q, \Sigma, \$, \tau, I, F, \delta)$ is an NFAwtl, then we can assign a stl-det-local-CD-R(1)-system $\mathcal{M}$ to it by defining $\mathcal{M} = ((M_j, \sigma_j)_{j \in J}, J_0)$ as follows. The set of indices is $J = Q \times \Sigma$, and $J_0 = I \times \Sigma$. For all pairs $(q, a) \in J$, $M_{(q,a)} = (\Sigma, \text{¢}, \$, 1, \delta_{(q,a)})$ is defined by the following transition relation and successor set:

$\delta_{(q,a)}(¢) = \mathsf{MVR}$, $\quad \delta_{(q,a)}(b) = \mathsf{MVR}$ for all $b \in \tau(q)$,
$\delta_{(q,a)}(\$) = \mathsf{Accept}$, if $q \in F$, $\quad \delta_{(q,a)}(a) = \varepsilon$, $\quad$ if $a \notin \tau(q)$ and $\delta(q,a) \neq \emptyset$,

$$\sigma_{(q,a)} = \begin{cases} \delta(q,a) \times \Sigma, \text{ if } \delta(q,a) \neq \emptyset, \\ Q \times \Sigma, \quad \text{otherwise.} \end{cases}$$

Further, $\delta_{(q,a)}(c) = \emptyset$ for all letters $c \in (\Sigma \setminus (\tau(q) \cup \{a\}))$, and $\delta_{(q,a)}(\$) = \emptyset$, if q is not a final state of A. Observe that the automaton $M_{(q,a)}$ cannot make any rewrite/restart step, and hence, its successor set is never used, if $a \in \tau(q)$ or $\delta(q,a) = \emptyset$.

A computation step $q_1uav \cdot \$ \vdash_A q_2uv \cdot \$$ of A is simulated by the component $M_{(q_1,a)}$ of $\mathcal{M}$ as $(\varepsilon, ¢ \cdot uav \cdot \$) \vdash^*_{M_{(q_1,a)}} (¢ \cdot u, av \cdot \$) \vdash_{M_{(q_1,a)}} (\varepsilon, ¢ \cdot uv \cdot \$)$. Further, an accepting step of A of the form $q_1u \cdot \$ \vdash_A \mathsf{Accept}$ is simulated by a component $M_{(q_1,a)}$ as $(\varepsilon, ¢ \cdot u \cdot \$) \vdash^*_{M_{(q_1,a)}} (¢ \cdot u, \$) \vdash_{M_{(q_1,a)}} \mathsf{Accept}$. Thus, in order to simulate an accepting computation of A, one must guess the next letter to be deleted in each step, and choose the corresponding component of $\mathcal{M}$. It now follows easily that $L(\mathcal{M}) = L(A)$ holds.

Conversely, if $\mathcal{M} = ((M_j, \sigma_j)_{j \in J}, J_0)$ is a CD-system of stateless deterministic R(1)-automata $M_j = (\Sigma, ¢, \$, 1, \delta_j)$ $(j \in J)$, then we can associate an NFAwtl $A = (J \cup \{+\}, \Sigma, \$, \tau, J_0, F, \delta)$ to $\mathcal{M}$ as follows. For each index $j \in J$, we define the translucency mapping τ and the transition function δ as follows:

$$\tau(j) = \begin{cases} \Sigma, & \text{if } \delta_j(¢) = \mathsf{Accept}, \\ \Sigma_1^{(j)}, & \text{otherwise,} \end{cases} \Bigg\}, \quad \begin{array}{l} \delta(j,a) = \sigma_j \quad \text{for all } a \in \Sigma_2^{(j)}, \\ \delta(j,b) = \{+\} \text{ for all } b \in \Sigma_3^{(j)}. \end{array}$$

Here $\Sigma_1^{(j)}$, $\Sigma_2^{(j)}$, and $\Sigma_3^{(j)}$ are the subsets of Σ mentioned above that correspond to the R(1)-automaton M_j. Further, we define $\tau(+) = \Sigma$ and

$$F = \{\, j \in J \mid \delta_j(\$) = \mathsf{Accept} \,\} \cup \{+\}.$$

It can now be verified that A can simulate the accepting computations of $\mathcal{M}$ in a stepwise fashion. Thus, it follows that $L(A) = L(\mathcal{M})$ holds.

Now Proposition 1 and Theorem 1 are immediate consequences of the above correspondence and the results on stl-det-local-CD-R(1)-systems presented in [7], while the closure and non-closure results as well as the decidability and undecidability results follow from [8].

References

1. Balan, M.S.: Automaton models inpsired by peptide computing. In: Domaratzki, M., Salomaa, K. (eds.): UC'07, Workshop on Language Theory in Biocomputing, Proc., Kingston, Ontario, Canada (2007) 1–15
2. Diekert, V., Rozenberg, G. (eds.): The Book of Traces, World Scientific, Singapore (1995)
3. Ibarra, O.: Reversal-bounded multicounter machines and their decision problems. J. Assoc. Comput. Mach. 25 (1978) 116–133
4. Jančar, P., Mráz, F., Plátek, M., Vogel, J.: Restarting automata. In: Reichel, H. (ed.): FCT 1995, Proc., Lecture Notes in Computer Science, Vol. 965, Springer-Verlag, Berlin Heidelberg (1995) 283–292

5. Joshi, A.K.: Mildly Context-Sensitive Grammars. http://www.kornai.com/MatLing/mcsfin.pdf (12.11.2010)
6. Mery, B., Amblard, M., Durand, I., Retoré, C.: A Case Study of the Convergence of Mildly Context-Sensitive Formalisms for Natural Language Syntax: from Minimalist Grammars to Multiple Context-Free Grammars. Rapport de recherche, nr. 6042, INRIA Futurs, Parc Club Orsay Université, Orsay (2006)
7. Nagy, B., Otto, F.: CD-systems of stateless deterministic R(1)-automata accept all rational trace languages. In: Dediu, A.H., Fernau, H., Martin-Vide, C. (eds.): LATA 2010, Proc., Lecture Notes in Computer Science, Vol. 6031, Springer-Verlag, Berlin Heidelberg (2010) 463–474
8. Nagy, B., Otto, F.: CD-Systems of Stateless Deterministic R-Automata with Window Size One. Kasseler Informatikschriften 2/2010, Fachbereich Elektrotechnik/Informatik, Universität Kassel, 2010. https://kobra.bibliothek.uni-kassel.de/handle/ urn:nbn:de:hebis:34-2010042732682