On CD-Systems of Stateless Deterministic R-Automata with Window Size One*

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

[1] Department of Computer Science, Faculty of Informatics
University of Debrecen
4032 Debrecen, Egyetem tér 1., Hungary
nagy.benedek@inf.unideb.hu

[2] Fachbereich Elektrotechnik/Informatik
Universität Kassel
34109 Kassel, Germany
otto@theory.informatik.uni-kassel.de

Abstract. Here we study cooperating distributed systems (CD-systems) of restarting automata that are very restricted: they are deterministic, they cannot rewrite, but only delete symbols, they restart immediately after performing a delete operation, they are stateless, and they have a read/write window of size 1 only, that is, these are stateless deterministic R(1)-automata. We study the expressive power of these systems by relating the class of languages that they accept by mode = 1 computations to other well-studied language classes, showing in particular that this class only contains semi-linear languages, and that it includes all rational trace languages. In addition, we investigate the closure and non-closure properties of this class of languages and some of its algorithmic properties.

1 Introduction

Cooperating distributed systems (CD-systems) of restarting automata have been defined in [18], and in [19, 20] various types of deterministic CD-systems of restarting automata have been studied. As expected CD-systems are much more expressive than their component automata themselves. For example, already the marked copy language $L_{\text{copy}} = \{ wcw \mid w \in \{a, b\}^* \}$ is accepted by a CD-system consisting of only two deterministic R-automata, although this language is not even growing context-sensitive [3, 14], that is, it is not even accepted by any deterministic RRWW-automaton. On the other hand, stateless restarting automata, that is, restarting automata with only a single state, have been introduced and studied in [11, 12]. In the monotone case and in the deterministic case, they are just as expressive as the corresponding restarting automata with states, provided that auxiliary symbols are available. Without the latter, however, stateless

* This work was supported by grants from the Balassi Intézet Magyar Ösztöndíj Bizottsága (MÖB) and the Deutsche Akademischer Austauschdienst (DAAD).

restarting automata are in general much less expressive than their corresponding counterparts with states.

Here we study deterministic restarting automata that are stateless and that have a read/write window of a fixed size $k > 0$, and CD-systems of such automata. In fact, we mainly concentrate on CD-systems of stateless deterministic R-automata with window size 1. The restarting automata of this type are really very restricted, and accordingly their expressive power is very limited. However, by combining several such automata into a CD-system we obtain a device that is suprizingly expressive, as we will see.

We first consider stateless deterministic R-automata, showing that we obtain an infinite hierarchy of language classes based on the window size. In fact, the different levels of this hierarchy can be separated from one another by regular languages. As already stateless deterministic R-automata of window size 2 can accept the Dyck language $D_n'^*$ for all $n \geq 1$ (see, e.g., [1]), this shows that, for all $k \geq 2$, the class $\mathcal{L}(\mathsf{stl\text{-}det\text{-}R}(k))$ of languages accepted by stateless deterministic R-automata of window size k is incomparable under inclusion to the class REG of regular languages. However, all regular languages are accepted by stateless deterministic R-automata. Further, each stateless deterministic R-automaton of window size 2 is necessarily monotone, which implies that it accepts a deterministic context-free language. On the other hand, the class $\mathcal{L}(\mathsf{stl\text{-}det\text{-}R}(9))$ contains a non-context-free language. Thus, for all $k \geq 9$, the class $\mathcal{L}(\mathsf{stl\text{-}det\text{-}R}(k))$ is incomparable under inclusion to the class CFL of context-free languages.

Then we restate the definition of CD-systems of restarting automata, and turn to our main topic, the CD-systems of stateless deterministic R(1)-automata. We compare the class of languages that are accepted by these systems through mode = 1 computations to other well-known language classes. In particular, we show that in mode = 1 these systems only accept languages with semi-linear Parikh image, including all regular languages, but that they also accept some languages that are not even context-free. In fact, these systems accept all rational trace languages. Accordingly they can also be interpreted as a refinement of the so-called *multiset finite automata* of [5], which accept all regular macrosets, that is, the commutative closures of all regular languages. In addition, we present a syntactic restriction for CD-systems of stateless deterministic R-automata of window size 1 such that the corresponding systems characterize the class of rational trace languages. These systems actually yield an effective calculus for rational trace languages in that from systems of this form for rational trace languages S_1 and S_2 we can effectively construct systems for the rational trace language $S_1 \cup S_2$, $S_1 \cdot S_2$, and S_1^*. Then we study closure and non-closure properties of the class of languages accepted by CD-systems of stateless deterministic R(1)-automata. We prove that this class is closed under union, product, Kleene-star, and inverse projections, but that it is neither closed under intersection with regular languages nor under ε-free morphisms. Finally we address some algorithmic problems for CD-systems of stateless deterministic R(1)-automata like the emptiness problem, the finiteness problem, and the equivalence problem. The paper closes with a short summary and some open problems for future work.

2 Stateless R-Automata with Constant Window Size

We first describe in short the types of restarting automata we will be dealing with. More details can be found in [24].

A *one-way restarting automaton*, abbreviated as RRWW-*automaton*, is a one-tape machine that is described by an 8-tuple $M = (Q, \Sigma, \Gamma, ¢, \$, q_0, k, \delta)$, where Q is a finite set of states, Σ is a finite input alphabet, Γ is a finite tape alphabet containing Σ, the symbols $¢, \$ \notin \Gamma$ serve as markers for the left and right border of the work space, respectively, $q_0 \in Q$ is the initial state, $k \geq 1$ is the size of the *read/write window*, and δ is the *transition relation* that associates a finite set of *transition steps* to each pair (q, u) consisting of a state $q \in Q$ and a possible contents u of the read/write window. There are four types of transition steps: *move-right steps* (MVR), which shift the window one position to the right and change the internal state, *rewrite steps*, which replace the content u of the read/write window by a shorter word, thereby also shortening the tape, and change the internal state, *restart steps* (Restart), which place the read/write window over the left end of the tape, and reset the internal state to the initial state q_0, and *accept steps* (Accept), which cause M to halt and accept.

A *configuration* of M is described by a string $\alpha q \beta$, where $q \in Q$, and either $\alpha = \varepsilon$ (the empty word) and $\beta \in \{¢\} \cdot \Gamma^* \cdot \{\$\}$ or $\alpha \in \{¢\} \cdot \Gamma^*$ and $\beta \in \Gamma^* \cdot \{\$\}$; here q represents the current state, $\alpha\beta$ is the current content of the tape, and it is understood that the head scans the first k symbols of β or all of β when $|\beta| \leq k$. A *restarting configuration* is of the form $q_0 ¢ w \$$, where $w \in \Gamma^*$; if $w \in \Sigma^*$, then $q_0 ¢ w \$$ is an *initial configuration*. By $\vdash_M$ we denote the single-step computation relation that M induces on the set of its configurations, and $\vdash_M^*$ denotes the reflexive transitive closure of $\vdash_M$.

The automaton M proceeds as follows. Starting from an initial configuration $q_0 ¢ w \$$, the window moves right until a configuration of the form $¢xquy\$$ is reached such that $\delta(q, u)$ contains a rewrite step that rewrites u to v, that is, $(p, v) \in \delta(q, u)$ for some state $p \in Q$ and some word $v \in \Gamma^*$ satisfying $|v| < |u|$. If this particular transition is now chosen, then the latter configuration is transformed into the configuration $¢xvpy\$$. Then M performs some more move-right steps until a restart step is executed, which then yields the restarting configuration $q_0 ¢ xvy \$$. This computation, which is called a *cycle*, is expressed as $w \vdash_M^c xvy$. A computation of M consists of a finite sequence of cycles that is followed by a *tail computation*, which consists of a sequence of move-right operations that may include a single rewrite step, and that is completed by either an accept step, or that reaches a configuration in which M cannot perform another transition step. In the former case we say that M accepts, while in the latter it rejects. A word $w \in \Gamma^*$ is *accepted* by M, if there is a computation of M which starts with the configuration $q_0 ¢ w \$$, and which finishes by executing an accept step. By $L_C(M)$ we denote the language consisting of all words accepted by M. It is called the *characteristic language* of M, and $L(M) = L_C(M) \cap \Sigma^*$ is the (*input*) *language* of M.

We are also interested in various restricted types of restarting automata. They are obtained by combining two types of restrictions:

(a) Restrictions on the movement of the read/write window (expressed by the first part of the class name): RR- denotes no restriction, and R- means that each rewrite step is combined with a restart operation.
(b) Restrictions on the rewrite-instructions (expressed by the second part of the class name): -WW denotes no restriction, -W means that no auxiliary symbols are available (that is, $\Gamma = \Sigma$), and -ε means that no auxiliary symbols are available and that each rewrite step is simply a deletion (that is, if M contains the rewrite operation $(p, v) \in \delta(q, u)$, then v is obtained from u by deleting some symbols).

In [11] the *stateless* variants of RWW-automata are studied, where an RWW-automaton $M = (Q, \Sigma, \Gamma, ¢, \$, q_0, k, \delta)$ is called *stateless* if $Q = \{q_0\}$ holds. Thus, in this case M can simply be described by the 6-tuple $M = (\Sigma, \Gamma, ¢, \$, k, \delta)$. In the original definition it was required that a stateless RWW-automaton may execute an accept instruction only at the right end of the tape, that is, when it sees the right delimiter \$, but this is actually just a convenience, as shown by the following proposition.

Proposition 1. [13] *Given a stateless* RWW*-automaton* $M = (\Sigma, \Gamma, ¢, \$, k, \delta)$*, one can construct a stateless* RWW*-automaton* $M' = (\Sigma, \Gamma, ¢, \$, k + 1, \delta')$ *that executes accept instructions only at the right end of the tape, and that accepts the same characteristic language as* M*. If* M *is an* RW*-automaton or an* R*-automaton, then so is* M'*, and if* M *is deterministic, then so is* M'*.*

In [11] the following results were obtained. Here the prefix stl- is used to denote stateless types of restarting automata, the prefix det- is used to denote deterministic types of restarting automata, and the prefix mon- is used to denote restarting automata that are monotone. Here a restarting automaton M is called *monotone*, if the distance from the place of rewriting to the right end of the tape does not increase from one cycle to the next in any computation of M. We use the notation $\mathcal{L}(\mathsf{X})$ to denote the class of (input) languages that are accepted by automata of type X.

Theorem 1. (a) $\mathcal{L}(\text{stl-det-mon-RWW}) = \text{DCFL}$.
(b) $\mathcal{L}(\text{stl-mon-RWW}) = \text{CFL}$.
(c) $\mathcal{L}(\text{stl-det-RWW}) = \text{CRL}$.
(d) $\mathcal{L}(\text{stl-det-mon-R}) \supsetneq \text{REG}$.

Here CRL denotes the class of *Church-Rosser languages* of McNaughton et. al. [16], DCFL is the class of deterministic context-free languages, and REG denotes the class of regular languages.

We are interested in stateless R-automata with a fixed window size. For each positive integer k, we denote by stl-det-R(k) the class of stateless deterministic

R-automata that have a read/write window of size k. We will see that there is an infinite hierarchy of language classes $\mathcal{L}(\mathsf{stl\text{-}det\text{-}R}(k))$ based on the value of the parameter k.

First we consider stateless deterministic R-automata with window size 1. For these automata we introduce the following notions that we will repeatedly use throughout the paper.

Definition 1. *Assume that $M = (\Sigma, \Sigma, ¢, \$, 1, \delta)$ is a stateless deterministic* R*-automaton of window size 1. Then we can partition the alphabet Σ into four disjoint subalphabets:*

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

Thus, Σ_1 is the set of letters that M just moves across, Σ_2 is the set of letters that M deletes, Σ_3 is the set of letters which cause M to accept, and Σ_4 is the set of letters on which M will get stuck.

Then the following characterization holds.

Proposition 2. *Let $M = (\Sigma, \Sigma, ¢, \$, 1, \delta)$ be a stateless deterministic* R(1)*-automaton, and assume that the subalphabets $\Sigma_1, \Sigma_2, \Sigma_3, \Sigma_4$ are defined as above. Then the simple language $S(M)$ of words accepted by M in tail computations is characterized as*

$$S(M) = \begin{cases} \Sigma^*, & \text{if } \delta(¢) = \mathsf{Accept},\\ \Sigma_1^* \cdot \Sigma_3 \cdot \Sigma^*, & \text{if } \delta(¢) = \mathsf{MVR} \text{ and } \delta(\$) \neq \mathsf{Accept},\\ \Sigma_1^* \cdot ((\Sigma_3 \cdot \Sigma^*) \cup \{\varepsilon\}), & \text{if } \delta(¢) = \mathsf{MVR} \text{ and } \delta(\$) = \mathsf{Accept}, \end{cases}$$

and the language $L(M)$ is characterized as

$$L(M) = \begin{cases} \Sigma^*, & \text{if } \delta(¢) = \mathsf{Accept},\\ (\Sigma_1 \cup \Sigma_2)^* \cdot \Sigma_3 \cdot \Sigma^*, & \text{if } \delta(¢) = \mathsf{MVR} \text{ and } \delta(\$) \neq \mathsf{Accept},\\ (\Sigma_1 \cup \Sigma_2)^* \cdot ((\Sigma_3 \cdot \Sigma^*) \cup \{\varepsilon\}), & \text{if } \delta(¢) = \mathsf{MVR} \text{ and } \delta(\$) = \mathsf{Accept}. \end{cases}$$

Proof. If $\delta(¢) = \mathsf{Accept}$, then obviously M accepts each word $w \in \Sigma^*$ in a tail computation. Thus, we can concentrate on the case that $\delta(¢) = \mathsf{MVR}$ holds. Obviously, M will then accept each word from $\Sigma_1^* \cdot \Sigma_3 \cdot \Sigma^*$ in a tail computation, and if $\delta(\$) = \mathsf{Accept}$, it will also accept each word from Σ_1^* in a tail computation. Further, each word $w = uav$, where $u \in \Sigma_1^*$, $a \in \Sigma_2$, and $v \in \Sigma^*$ will cause a cycle of the form $w = uav \vdash_M^c uv$. Hence, one by one those letters from Σ_2 are removed from w that in w are only preceded by letters from $\Sigma_1 \cup \Sigma_2$. This yields the above description for the language $L(M)$. □

It is easily seen that a stateless finite-state acceptor with input alphabet Σ accepts a language of the form Σ_0^*, where Σ_0 is a subalphabet of Σ. Thus, we have the following easy consequence.

Corollary 1. *A language L is accepted by a stateless deterministic* R(1)*-automaton that only accepts on reaching the right delimiter \$, if and only if L is the simple language of a stateless deterministic* R(1)*-automaton that only accepts on reaching the right delimiter \$, if and only if L is accepted by a stateless finite-state acceptor.*

Proof. Let $M = (\Sigma, \Sigma, ¢, \$, 1, \delta)$ be a stateless deterministic R(1)-automaton that only accepts on reaching the right delimiter \$. Then $\Sigma_3 = \emptyset$, and hence we see from the above proposition that $L(M) = \Sigma^*$, if $\delta(¢) = \mathsf{Accept}$, and that $L(M) = (\Sigma_1 \cup \Sigma_2)^*$, otherwise. On the other hand, if A is a stateless finite-state acceptor on Σ that accepts the language Σ_0^*, then we obtain a stateless deterministic R(1)-automaton $M = (\Sigma, \Sigma, ¢, \$, 1, \delta)$ by defining $\delta(¢) = \mathsf{MVR}$, $\delta(a) = \mathsf{MVR}$ for all letters $a \in \Sigma_0$, and $\delta(\$) = \mathsf{Accept}$. Then $L(M) = S(M) = \Sigma_0^*$. □

Thus, stateless deterministic R(1)-automata can be seen as stateless deterministic finite-state acceptors that are enabled to accept without having read their input completely.

Next we turn to stateless deterministic R-automata of window size 2.

Lemma 1. *The Dyck language $D_n'^*$ is accepted by a stateless deterministic* R(2)*-automaton for each integer $n \geq 1$.*

Proof. The Dyck language $D_1'^*$ is defined over the alphabet $T_1 = \{a, b\}$. It is generated by the context-free grammar $G_1 = (\{S, A\}, T_1, P_1, S)$, where P_1 contains the following productions:

$$S \to AS,\ S \to \varepsilon,\ A \to aSb.$$

In fact, a word $w \in T_1^*$ belongs to $D_1'^*$ if and only if $|w|_a = |w|_b$, and for each proper prefix x of w we have $|x|_a \geq |x|_b$. Thus, it is easily seen that $D_1'^*$ is accepted by the stateless deterministic R(2)-automaton M_1 that is defined by the following transition function:

$$\begin{array}{ll} (1.)\ \delta(¢\$) = \mathsf{Accept}, & (3.)\ \delta(aa) = \mathsf{MVR}, \\ (2.)\ \delta(¢a) = \mathsf{MVR}, & (4.)\ \delta(ab) = \varepsilon. \end{array}$$

Thus, we see that $D_1'^* \in \mathcal{L}(\mathsf{stl\text{-}det\text{-}R(2)})$ holds. It can be shown analogously that each Dyck language $D_n'^*$, $n \geq 2$, is accepted by a stateless deterministic R(2)-automaton. □

If $M = (\Sigma, \Sigma, ¢, \$, 2, \delta)$ is a stateless deterministic R(2)-automaton, then each cycle $w \vdash_M^c w'$ has the form $w = uabv$ and $w' = ucv$, where $u, v \in \Sigma^*$, $a, b \in \Sigma$, and $c \in \Sigma \cup \{\varepsilon\}$. As M is deterministic, it must scan the prefix uc of w' completely before it can apply another delete step. Hence, we see that M is necessarily monotone. As monotone deterministic R-automata accept deterministic context-free languages only (see, e.g., [24]), this observation has the following consequence.

Lemma 2. $\mathcal{L}(\mathsf{stl\text{-}det\text{-}R}(2)) \subseteq \mathsf{DCFL}$.

Actually the above inclusion is a proper one. This follows immediately from the following result.

Lemma 3. *For each integer $k \geq 1$, there exists a regular language $L_k \subseteq \{a,b\}^*$ such that $L_k \in \mathcal{L}(\mathsf{stl\text{-}det\text{-}R}(k+1)) \setminus \mathcal{L}(\mathsf{stl\text{-}det\text{-}R}(k))$, that is, L_k is accepted by a stateless deterministic* R*-automaton of window size $k+1$, but it is not accepted by any stateless deterministic* R*-automaton of window size k.*

Proof. For $k \geq 1$, let $L_k = \{\, (ab^k)^i \mid i \geq 0 \,\}$. Then L_k is obviously regular. Further, it is accepted by the stateless deterministic R-automaton of window size $k+1$ that is defined by the following transition function δ_{k+1}:

$$(1.)\ \delta_{k+1}(¢\$) = \mathsf{Accept}, \quad (2.)\ \delta_{k+1}(¢ab^{k-1}) = \mathsf{MVR}, \quad (3.)\ \delta_{k+1}(ab^k) = \varepsilon.$$

On the other hand, if $M = (\Sigma, \Sigma, ¢, \$, k, \delta)$ is a stateless deterministic R-automaton of window size k only that accepts the language L_k, then on input ab^kab^k, M will have to accept. However, $\delta(¢ab^{k-2})$ can neither be an accept nor a delete operation, and so $\delta(¢ab^{k-2}) = \mathsf{MVR}$. Thus, after the first step M reaches the configuration $¢qab^kab^k\$$, where q symbolizes the unique state of M. Now $\delta(ab^{k-1})$ has to be applied. Again it can neither be an accept nor a delete operation, that is, $\delta(ab^{k-1}) = \mathsf{MVR}$, and so M reaches the configuration $¢aqb^kab^k\$$. Continuing in this way we see that M will just move across its tape inscription, that is, $\delta(z) = \mathsf{MVR}$ for all $z \in \{ab^{k-1}, b^k, b^{k-1}a, b^{k-2}ab, \ldots, bab^{k-2}\}$. Finally M will reach the configuration $¢ab^kabqb^{k-1}\$$, and it will have to accept. However, it will then also accept the word $ab^{k+1}ab^k$ that does not belong to the language L_k. Hence, L_k is not accepted by any stateless deterministic R-automaton of window size k. □

This yields the following infinite hierarchy.

Corollary 2. *The language classes $(\mathcal{L}(\mathsf{stl\text{-}det\text{-}R}(k)))_{k\geq 1}$ form an infinite strictly increasing sequence. For all $k \geq 2$, the class $\mathcal{L}(\mathsf{stl\text{-}det\text{-}R}(k))$ is incomparable under inclusion to the class* REG *of regular languages.*

Stateless deterministic R-automata of window size 2 only accept certain deterministic context-free languages. Next we will see that with larger window size these automata do even accept some languages that are not context-free.

Let L_{expo} and $L_{\text{expo}}^{(\varphi)}$ be the following languages over $\{a,b\}$:

$$L_{\text{expo}} = \{\, a^{i_0}ba^{i_1}b\cdots a^{i_{n-1}}ba^{i_n} \mid n \geq 0,\ i_0,\ldots,i_n \geq 0,\ \text{and} \\ \exists m \geq 0 : \textstyle\sum_{j=0}^{n} 2^j \cdot i_j = 2^m \,\} \cup b^*,\ \text{and}$$
$$L_{\text{expo}}^{(\varphi)} = \varphi(L_{\text{expo}}),$$

where φ is the morphism induced by $a \mapsto ab$ and $b \mapsto b$. These languages are not context-free, as $L_{\text{expo}} \cap a^* = \{\, a^{2^n} \mid n \geq 0 \,\}$ and $L_{\text{expo}}^{(\varphi)} \cap (ab)^* = \{\, (ab)^{2^n} \mid n \geq 0 \,\}$.

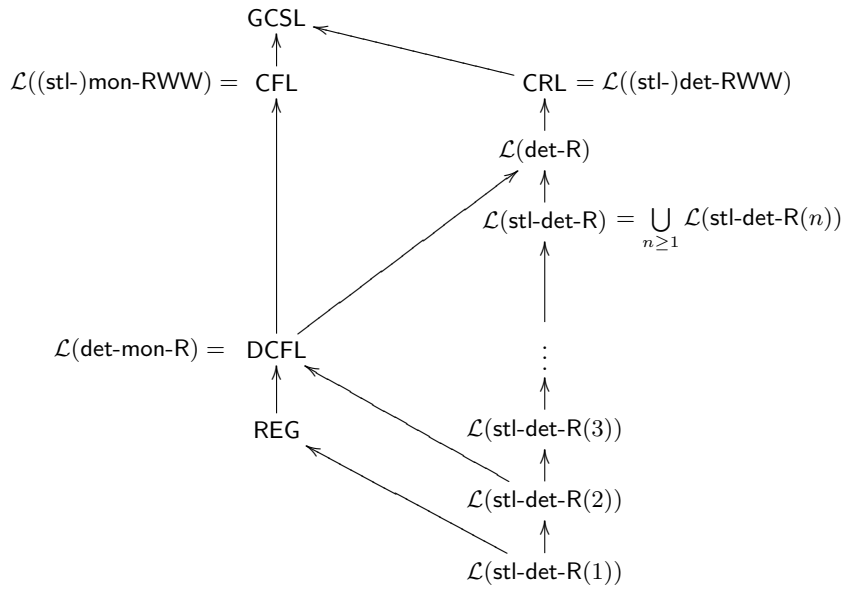

Figure 1. Taxonomy of language classes accepted by stateless deterministic R-automata. Here an arrow indicates a proper inclusion, and GCSL denotes the class of growing context-sensitive languages.

On the other hand, it is shown in [13] that the language $L_{\text{expo}}^{(\varphi)}$ is accepted by a stateless determinsitic R-automaton. In fact, the particular R-automaton for this language that is presented there has window size 9. This yields the following consequence.

Corollary 3. *For all $k \geq 9$, the class $\mathcal{L}$(stl-det-R(k)) is incomparable under inclusion to the class* CFL *of context-free languages.*

Open Problem 1. *What is the smallest integer k such that the class $\mathcal{L}$(stl-det-R(k)) contains a non-context-free language? From our results above we know that $3 \leq k \leq 9$ holds, but it is open whether already the class $\mathcal{L}$(stl-det-R(3)) contains a non-context-free language.*

In [11] it is shown that the deterministic linear language

$$L_d = \{\, ca^nb^n \mid n \geq 0 \,\} \cup \{\, da^nb^{2n} \mid n \geq 0 \,\}$$

is not accepted by any stateless RW-automaton. This yields the following non-inclusion result.

Corollary 4. DCFL $\not\subseteq \mathcal{L}$(stl-det-R).

Hence, the class $\mathcal{L}$(stl-det-R) is incomparable to the class of (deterministic) context-free languages. The diagram in Figure 1 summarizes the inclusion re-

lations between the language classes that are accepted by the various types of stateless deterministic R-automata and some classical language families.

3 CD-Systems of Restarting Automata

Here we restate the definition of a CD-system of restarting automata from [18] in short.

A *cooperating distributed system of* RRWW-*automata* (or a CD-RRWW-system, for short) consists of a finite collection $\mathcal{M} = ((M_i, \sigma_i)_{i \in I}, I_0)$ of RRWW-automata $M_i = (Q_i, \Sigma, \Gamma_i, \mathfrak{c}, \$, q_0^{(i)}, k, \delta_i)$ $(i \in I)$, *successor relations* $\sigma_i \subseteq I$ $(i \in I)$, and a subset $I_0 \subseteq I$ of *initial indices*. Here it is required that $Q_i \cap Q_j = \emptyset$ for all $i, j \in I$, $i \neq j$, that $I_0 \neq \emptyset$, that $\sigma_i \neq \emptyset$ for all $i \in I$, and that $i \notin \sigma_i$ for all $i \in I$.

Various modes of operation like $= j$, $\leq j$, $\geq j$ for $j \geq 1$ and t have been introduced and studied, but here we are only interested in mode $= 1$ computations. The computation of $\mathcal{M}$ in mode $= 1$ on an input word w proceeds as follows. First an index $i_0 \in I_0$ is chosen nondeterministically. Then the RRWW-automaton M_{i_0} starts the computation with the initial configuration $q_0^{(i_0)}\mathfrak{c}w\$$, and executes one cycle. Thereafter an index $i_1 \in \sigma_{i_0}$ is chosen nondeterministically, and M_{i_1} continues the computation by executing one cycle. This continues until, for some $l \geq 0$, the machine M_{i_l} accepts. Should at some stage the chosen machine M_{i_l} be unable to execute a cycle or to accept, then the computation fails.

By $L_{=1}(\mathcal{M})$ we denote the language that the CD-RRWW-system $\mathcal{M}$ accepts in mode $= 1$. It consists of all words $w \in \Sigma^*$ that are accepted by $\mathcal{M}$ in mode $= 1$ as described above. If X is any of the above types of restarting automata, then a CD-X-system is a CD-RRWW-system for which all component automata are of type X.

A CD-system of restarting automata $\mathcal{M} = ((M_i, \sigma_i)_{i \in I}, I_0)$ is called *stateless* if all component automata M_i $(i \in I)$ are stateless. Here we are interested in CD-systems of stateless deterministic R-automata. For these systems we use the notation stl-det-local-CD-R in accordance with the notation introduced in [20]. Observe that the computations of such a CD-system are not completely deterministic, as the starting component and the respective successor components are still chosen nondeterministically from among all available component automata. By $\mathcal{L}_{=1}(\text{stl-det-local-CD-R}(i))$ we denote the class of languages that are accepted by mode $= 1$ computations of stl-det-local-CD-R-systems with window size i. The following example illustrates the expressive power of these systems.

Example 1. We consider the *marked copy language* $L_{\text{copy}} = \{\, wcw \mid w \in \{a, b\}^* \,\}$ on $\Sigma = \{a, b, c\}$. It is well-known that this language is not even growing context-sensitive (see, e.g., [24]), and so it is not accepted by any deterministic RRWW-automaton. However, we will see that it is accepted by a stl-det-local-CD-R(2)-system with four components working in mode $= 1$.

Let $\mathcal{M} = ((M_i, \sigma_i)_{i \in I}, I_0)$, where $I = \{a, b, -, +\}$, $I_0 = \{a, b, +\}$, $\sigma_a = \{-\} = \sigma_b$, $\sigma_- = \{a, b, +\}$, $\sigma_+ = \{-\}$, and M_a, M_b, M_-, and M_+ are the stateless deterministic $\mathsf{R}(2)$-automata that are given by the following transition functions:

$$\begin{aligned}
M_a &: (1.)\ \delta_a(\mathbb{c}a) = \mathsf{MVR}, \\
&\quad (2.)\ \delta_a(xy) = \mathsf{MVR} \quad \text{for all } x \in \{a, b\} \text{ and } y \in \Sigma, \\
&\quad (3.)\ \delta_a(ca) = c, \\
M_b &: (4.)\ \delta_b(\mathbb{c}b) = \mathsf{MVR}, \\
&\quad (5.)\ \delta_b(xy) = \mathsf{MVR} \quad \text{for all } x \in \{a, b\} \text{ and } y \in \Sigma, \\
&\quad (6.)\ \delta_b(cb) = c, \\
M_- &: (7.)\ \delta_-(\mathbb{c}x) = \mathbb{c} \quad \text{for all } x \in \{a, b\}, \\
M_+ &: (8.)\ \delta_+(\mathbb{c}c) = \mathsf{MVR}, \\
&\quad (9.)\ \delta_+(c\$) = \mathsf{Accept}.
\end{aligned}$$

Obviously $\mathcal{M}$ accepts all words $z \in L_{\text{copy}}$ working in mode = 1. On the other hand, if a word $z \in \Sigma^*$ is accepted by $\mathcal{M}$ in mode = 1, then $z = wcw$ for some $w \in \{a, b\}^*$. It follows that $L_{=1}(\mathcal{M}) = L_{\text{copy}}$ holds, which implies that $L_{\text{copy}} \in \mathcal{L}_{=1}(\mathsf{stl\text{-}det\text{-}local\text{-}CD\text{-}R(2)})$. Thus, already the language class $\mathcal{L}_{=1}(\mathsf{stl\text{-}det\text{-}local\text{-}CD\text{-}R(2)})$ contains languages that are not even growing context-sensitive.

4 CD-Systems of Stateless Deterministic R-Automata with Window Size 1

As already CD-systems of stateless deterministic R-automata of window size 2 can accept some languages that are not even growing context-sensitive, we now concentrate on a class of CD-systems of restarting automata that are still more restricted: CD-systems of stateless deterministic R-automata of window size 1. As shown by Proposition 2 stateless deterministic R-automata of window size 1 can only accept regular languages of a rather restricted form. So it is certainly of interest to investigate the expressive power of CD-systems of restarting automata of this very restricted form. We start our investigation by presenting two examples of non-regular languages that are accepted by CD-systems of this form.

Proposition 3. *The Dyck language $D_1'^*$ is accepted by a CD-system of stateless deterministic* R*-automata of window size 1 working in mode = 1.*

Proof. Let $\mathcal{M} = ((M_i, \sigma_i)_{i \in I}, I_0)$, where $I = \{a, b, +\}$, $I_0 = \{a, +\}$, $\sigma_a = \{b\}$, $\sigma_b = \{a, +\}$, $\sigma_+ = \{a\}$, and M_a, M_b, and M_+ are the stateless deterministic R-automata of window size 1 that are given by the following transition functions:

$$\begin{aligned}
M_a &: (1.)\ \delta_a(\mathbb{c}) = \mathsf{MVR}, \\
&\quad (2.)\ \delta_a(a) = \varepsilon,
\end{aligned}$$

$$\begin{aligned}
M_b \;:\; &(3.)\ \delta_b(¢) = \mathsf{MVR},\\
&(4.)\ \delta_b(a) = \mathsf{MVR},\\
&(6.)\ \delta_b(b) = \varepsilon,\\
M_+ : &(10.)\ \delta_+(¢) = \mathsf{MVR},\\
&(11.)\ \delta_+(\$) = \mathsf{Accept}.
\end{aligned}$$

Let $w \in \{a,b\}^*$ be given as input. The automaton M_+ accepts the empty word and rejects (that is, gets stuck on) all other inputs. As $+ \in I_0$, we see that the empty word is accepted by $\mathcal{M}$ working in mode $= 1$. If $w \neq \varepsilon$, then the computation starts with M_a. If $w = aw_1$, then M_a simply deletes the first occurrence of a in w, otherwise, it gets stuck. Then M_b takes over, which deletes the first occurrence of the letter b, provided $|w_1|_b > 0$. Now this sequence consisting of two cycles is repeated until either the empty word is reached, and then the computation finishes with M_+ accepting, or until a non-empty word is reached that does not start with the letter a, or that does not contain any occurrences of the letter b, and then the computation gets stuck. It follows that $L_{=1}(\mathcal{M}) = D_1'^*$ holds. □

Proposition 4. *The language* $L_{abc} = \{\, w \in \{a,b,c\}^* \mid |w|_a = |w|_b = |w|_c \geq 0 \,\}$ *is accepted by a CD-system of stateless deterministic* R*-automata of window size 1 working in mode* $= 1$.

Proof. Let $\mathcal{M} = ((M_i, \sigma_i)_{i \in I}, I_0)$, where $I = \{a,b,c,+\}$, $I_0 = \{a,+\}$, $\sigma_a = \{b\}$, $\sigma_b = \{c\}$, $\sigma_c = \{a,+\}$, $\sigma_+ = \{a\}$, and M_a, M_b, M_c, and M_+ are the stateless deterministic R-automata of window size 1 that are given by the following transition functions:

$$\begin{aligned}
M_a \;:\; &(1.)\ \delta_a(¢) = \mathsf{MVR},\\
&(2.)\ \delta_a(x) = \mathsf{MVR} \quad \text{for all } x \in \{b,c\},\\
&(3.)\ \delta_a(a) = \varepsilon,\\
M_b \;:\; &(4.)\ \delta_b(¢) = \mathsf{MVR},\\
&(5.)\ \delta_b(x) = \mathsf{MVR} \quad \text{for all } x \in \{a,c\},\\
&(6.)\ \delta_b(b) = \varepsilon,\\
M_c \;:\; &(7.)\ \delta_c(¢) = \mathsf{MVR},\\
&(8.)\ \delta_c(x) = \mathsf{MVR} \quad \text{for all } x \in \{a,b\},\\
&(9.)\ \delta_c(c) = \varepsilon,\\
M_+ : &(10.)\ \delta_+(¢) = \mathsf{MVR},\\
&(11.)\ \delta_+(\$) = \mathsf{Accept}.
\end{aligned}$$

Let $w \in \{a,b,c\}^*$ be given as input. The automaton M_+ accepts the empty word and rejects (that is, gets stuck on) all other inputs. As $+ \in I_0$, we see that the empty word is accepted by $\mathcal{M}$ working in mode $= 1$. If $w \neq \varepsilon$, then the computation starts with M_a. If $|w|_a > 0$, then M_a simply deletes the first occurrence of a in w, otherwise, it gets stuck. Then M_b takes over, which deletes the first occurrence of the letter b, provided $|w|_b > 0$. Finally M_c deletes the first

occurrence of the letter c, if $|w|_c > 0$. Now this sequence consisting of three cycles is repeated until either the empty word is reached, and then the computation finishes with M_+ accepting, or until a non-empty word is reached that does not contain occurrences of all three letters, and then the computation gets stuck. It follows that $L_{=1}(\mathcal{M}) = L_{abc}$ holds. □

Observe that the CD-system above for accepting the language L_{abc} consists of only four $\mathsf{R}(1)$-automata. As the language L_{abc} is not context-free, we have the following consequence.

Corollary 5. *The language class* $\mathcal{L}_{=1}(\mathsf{stl\text{-}det\text{-}local\text{-}CD\text{-}R}(1))$ *contains languages that are not context-free.*

On the other hand, all regular languages are accepted by $\mathsf{stl\text{-}det\text{-}local\text{-}CD\text{-}R}(1)$-systems working in mode $= 1$.

Proposition 5. $\mathsf{REG} \subsetneq \mathcal{L}_{=1}(\mathsf{stl\text{-}det\text{-}local\text{-}CD\text{-}R}(1))$.

Proof. Let $L \subseteq \Sigma^*$ be a regular language, and let $A = (Q, \Sigma, p_0, F, \delta)$ be a complete deterministic finite-state acceptor for L. From A we construct a $\mathsf{stl\text{-}det\text{-}local\text{-}CD\text{-}R}(1)$-system $\mathcal{M} = ((M_i, \sigma_i)_{i \in I}, I_0)$ as follows:

- The set of indices is $I = (Q \times \Sigma) \cup (Q' \times \Sigma) \cup \{+\}$, where $Q' = \{\, q' \mid q \in Q \,\}$ is a copy of Q such that $Q \cap Q' = \emptyset$,
- the set of initial indices is $I_0 = \begin{cases} \{\, (p_0, a) \mid a \in \Sigma \,\}, & \text{if } \varepsilon \notin L, \\ \{\, (p_0, a) \mid a \in \Sigma \,\} \cup \{+\}, & \text{if } \varepsilon \in L, \end{cases}$
- the successor relations are defined by
  - $\sigma_{(q,a)} = \begin{cases} \{\, (\delta(q,a), b) \mid b \in \Sigma \,\} \cup \{+\}, & \text{if } \delta(q,a) \neq q \text{ and } \delta(q,a) \in F, \\ \{\, (\delta(q,a), b) \mid b \in \Sigma \,\}, & \text{if } \delta(q,a) \neq q \text{ and } \delta(q,a) \notin F, \\ \{\, (q', b) \mid b \in \Sigma \,\} \cup \{+\}, & \text{if } \delta(q,a) = q \text{ and } q \in F, \\ \{\, (q', b) \mid b \in \Sigma \,\}, & \text{if } \delta(q,a) = q \text{ and } q \notin F, \end{cases}$
  - $\sigma_{(q',a)} = \begin{cases} \{\, (\delta(q,a), b) \mid b \in \Sigma \,\} \cup \{+\}, & \text{if } \delta(q,a) \in F, \\ \{\, (\delta(q,a), b) \mid b \in \Sigma \,\}, & \text{if } \delta(q,a) \notin F, \end{cases}$
  - $\sigma_+ = \{\, (p_0, a) \mid a \in \Sigma \,\}$,
- and the $\mathsf{stl\text{-}det\text{-}R}(1)$-automata $M_{(q,a)}$, $M_{(q',a)}$, and M_+ are defined by the following transition functions:

$$\begin{aligned} M_{(q,a)} \;:\; \delta_{(q,a)}(¢) &= \mathsf{MVR}, \\ \delta_{(q,a)}(a) &= \varepsilon, \\ M_{(q',a)} \;:\; \delta_{(q',a)}(¢) &= \mathsf{MVR}, \\ \delta_{(q',a)}(a) &= \varepsilon, \\ M_+ \;:\; \delta_+(¢) &= \mathsf{MVR}, \\ \delta_+(\$) &= \mathsf{Accept}. \end{aligned}$$

Then it can be checked easily that the accepting mode = 1 computations of $\mathcal{M}$ correspond one-to-one to the accepting computations of the finite-state acceptor A. In fact, if A executes the transition $\delta(q,a) = p$, then the component automaton $M_{(q,a)}$ (or $M_{(q',a)}$) must be active. It simply deletes the first letter to the right of the left delimiter ¢ (provided that is an a), and then the component automaton $M_{(p,b)}$ (or $M_{(p',b)}$, if $p = q$) becomes active, where it is guessed that the next letter to be processed by A is a b. Thus, it follows that $L = L(A) = L_{=1}(\mathcal{M})$ holds. □

Observe that the proof above crucially depends on the fact that in a mode = 1 computation of a stl-det-local-CD-R(1)-system, the initial component automaton and the successor automata are chosen nondeterministically from among the corresponding sets.

Open Problem 2. *Observe that the above simulation of a deterministic finite-state acceptor by a* stl-det-local-CD-R(1)*-system is rather inefficient, as we have used $O(|Q| \cdot |\Sigma|)$ many component automata. Is there a more efficient (that is, more succinct) simulation?*

Currently we have no answer to this question, but we can at least show that in some instances stl-det-local-CD-R(1)-systems are much more succinct than even nondeterministic finite-state acceptors. Here we take the number of component automata of a CD-system as its (static) complexity measure.

Example 2. Let $\Sigma = \{a, b, c\}$, and let $n \geq 1$. We define the language $L_{=n} \subseteq \Sigma^*$ as follows:

$$L_{=n} = \{\, w \in \Sigma^* \mid |w|_a = n = |w|_b \,\}.$$

We can easily construct a stl-det-local-CD-R(1)-system $\mathcal{M}$ with $2n + 1$ components that accepts the language $L_{=n}$ in mode = 1. We just need n component automata that each simple delete one occurrence of the letter a, while moving right across occurrences of the letters b and c, we need another n component automata that each simply delete one occurrence of the letter b, while moving right across occurrences of the letter c, and we need a final component that accepts all words from c^*.

Now assume that $A = (Q, \Sigma, q_0, F, \delta)$ is a nondeterministic finite-state acceptor for $L_{=n}$. We claim that A has at least $(n+1)^2$ many states. Just consider the words $x_{i,j} = a^i b^j$ and $y_{i,j} = a^{n-i} b^{n-j}$ for all $i, j = 0, 1, \ldots, n$. Then $x_{i,j} y_{i,j} = a^i b^j a^{n-i} b^{n-j} \in L_{=n}$ for all i, j, while $x_{i,j} y_{i',j'} \notin L_{=n}$, whenever $i' \neq i$ or $j' \neq j$. Thus, the set of pairs $(x_{i,j}, y_{i,j})_{i,j=0,\ldots,n}$ is a fooling set for $L_{=n}$. Accordingly it follows that $|Q| \geq (n+1)^2$ [2].

Analogously for the finite language

$$L'_{=n} = \{\, w \in \Sigma^* \mid |w|_a = |w|_b = |w|_c = n \,\}$$

we have a stl-det-local-CD-R(1)-system consisting of $3n+1$ component automata, while an NFA for this language needs at least $(n+1)^3$ many states.

Open Problem 3. *Can we realize an exponential trade-off between* stl-det-local-CD-R(1)*-systems and nondeterministic finite-state acceptors?*

Before continuing with the discussion of the properties of the language class $\mathcal{L}_{=1}(\text{stl-det-local-CD-R(1)})$, we introduce a normal form for stl-det-local-CD-R(1)-systems.

Definition 2. *A* stl-det-local-CD-R(1)*-system* $\mathcal{M} = ((M_i, \sigma_i)_{i \in i}, I_0)$ *on alphabet* Σ *is in* normal form*, if it satisfies the following three conditions for all* $i \in I$*, where* $\Sigma_1^{(i)}$*,* $\Sigma_2^{(i)}$*,* $\Sigma_3^{(i)}$*,* $\Sigma_4^{(i)}$ *is the partitioning of alphabet* Σ *from Definition 1 for automaton* M_i*:*

1. *For the component automaton* M_i*, we have* $|\Sigma_2^{(i)}| \leq 1$*, that is, there is at most one letter that* M_i *deletes.*
2. *All accept instructions are executed on the \$-symbol only, that is,* $\delta_i(¢) = \mathsf{MVR}$ *and* $\Sigma_3^{(i)} = \emptyset$*.*
3. M_i *does not have both, rewrite instructions and accept instructions, that is, if* $\delta_i(\$) = \mathsf{Accept}$*, then* $\Sigma_2^{(i)} = \emptyset$*.*

If $\mathcal{M}$ is in normal form, and $\Sigma_i^{(2)} = \emptyset$ and $\delta_i(\$) \neq \mathsf{Accept}$ for some index i, then M_i cannot be used in any accepting computation of $\mathcal{M}$, that is, we could simply drop M_i from $\mathcal{M}$. Hence, we can assume that $\delta_i(\$) = \mathsf{Accept}$ if and only if $\Sigma_i^{(2)} = \emptyset$.

Lemma 4. *From a* stl-det-local-CD-R(1)*-system* $\mathcal{M}$ *one can construct a* stl-det-local-CD-R(1)*-system* $\mathcal{M}'$ *in normal form such that* $L_{=1}(\mathcal{M}') = L_{=1}(\mathcal{M})$*.*

Proof. Let $\mathcal{M} = ((M_i, \sigma_i)_{i \in I}, I_0)$ be a stl-det-local-CD-R(1)-system. First we split every component automaton M_i into $|\Sigma_2^{(i)}| + 1$ many parts, $M_i^{(a)}$ for $a \in \Sigma_2^{(i)}$, and $M_i^{(+)}$, where the former is responsible for executing the cycles of M_i in which an occurrence of the letter a is deleted, while the latter takes care of the accepting tail computations of M_i. In detail, for each $a \in \Sigma_2^{(i)}$, and all $b, c \in \Sigma$,

$$
\begin{array}{ll}
\delta_i^{(a)} = \emptyset, \quad \text{if } \delta_i(¢) = \mathsf{Accept}, & \delta_i^{(+)}(¢) = \mathsf{Accept}, \text{ if } \delta_i(¢) = \mathsf{Accept}, \\
\delta_i^{(a)}(¢) = \mathsf{MVR}, \text{ if } \delta_i(¢) = \mathsf{MVR}, & \delta_i^{(+)}(¢) = \mathsf{MVR}, \quad \text{if } \delta_i(¢) = \mathsf{MVR}, \\
\delta_i^{(a)}(b) = \mathsf{MVR}, \text{ if } \delta_i(b) = \mathsf{MVR}, & \delta_i^{(+)}(b) = \mathsf{MVR}, \quad \text{if } \delta_i(b) = \mathsf{MVR}, \\
\delta_i^{(a)}(a) = \varepsilon, & \delta_i^{(+)}(c) = \mathsf{Accept}, \text{ if } \delta_i(c) = \mathsf{Accept}, \\
 & \delta_i^{(+)}(\$) = \mathsf{Accept}, \text{ if } \delta_i(\$) = \mathsf{Accept}.
\end{array}
$$

Then we adjust the successor relations σ_i ($i \in I$) as follows:

$$\sigma_i^{(a)} = \sigma_i^{(+)} = \{\, j^{(b)}, j^{(+)} \mid j \in \sigma_i,\, b \in \Sigma_2^{(j)} \,\}.$$

Observe, however, that the successor relations $\sigma_i^{(+)}$ are never used in any computation. Finally, we take $\hat{\mathcal{M}} = ((M_i^{(a)}, \sigma_i^{(a)})_{i \in I, a \in \Sigma_2^{(i)}} \cup (M_i^{(+)}, \sigma_i^{(+)})_{i \in I}), \hat{I}_0)$, where $\hat{I}_0 = \{\, i^{(a)}, i^{(+)} \mid i \in I_0,\, a \in \Sigma_2^{(i)} \,\}$.

Then $\hat{\mathcal{M}}$ simply simulates the computations of $\mathcal{M}$. Each time a successor automaton M_j is chosen in a computation of $\mathcal{M}$, one has to guess whether another cycle will be executed, and if so, which rewrite instruction will be applied, or whether the next component automaton will accept in a tail computation. Then in the simulating computation of $\hat{\mathcal{M}}$, one must simply choose the corresponding component $M_j^{(a)}$ or $M_j^{(+)}$. It follows easily that $L_{=1}(\hat{\mathcal{M}}) = L_{=1}(\mathcal{M})$.

In order to obtain the intended system in normal form, we modify the accepting component automata $M_i^{(+)}$ $(i \in I)$. Actually we need to distinguish three cases.

If $\delta_i^{(+)}(¢) = \mathsf{Accept}$, then $M_i^{(+)}$ will accept all words from Σ^*. Accordingly, we define $\delta_i'^{(+)}$ as follows:

$$\delta_i'^{(+)}(¢) = \mathsf{MVR},\ \delta_i'^{(+)}(a) = \mathsf{MVR} \text{ for all } a \in \Sigma,\ \delta_i'^{(+)}(\$) = \mathsf{Accept}.$$

Then $M_i'^{(+)}$ accepts all words from Σ^*, but it executes an accept instruction only on the \$-symbol.

If $\delta_i^{(+)}(¢) = \mathsf{MVR}$, and $\delta_i^{(+)}(\$)$ is undefined, then $M_i^{(+)}$ accepts all words from ${\Sigma_1^{(i)}}^* \cdot \Sigma_3^{(i)} \cdot \Sigma^*$. Accordingly, we define $\delta_i'^{(+)}$ as follows:

$$\begin{aligned} &\delta_i'^{(+)}(¢) = \mathsf{MVR}, \\ &\delta_i'^{(+)}(a) = \mathsf{MVR} \quad \text{for all } a \in \Sigma_1^{(i)}, \\ &\delta_i'^{(+)}(a) = \varepsilon \qquad \text{for all } a \in \Sigma_3^{(i)}. \end{aligned}$$

Also we define another component automaton $M_i''^{(+)}$ as follows:

$$\begin{aligned} &\delta_i''^{(+)}(¢) = \mathsf{MVR}, \\ &\delta_i''^{(+)}(a) = \mathsf{MVR} \quad \text{for all } a \in \Sigma, \\ &\delta_i'^{(+)}(\$) = \mathsf{Accept}, \end{aligned}$$

where $M_i''^{(+)}$ is the only successor of $M_i'^{(+)}$. Then together they accept the same words as $M_i^{(+)}$, but an accept instruction is only executed on the \$-symbol.

Finally, if $\delta_i^{(+)}(¢) = \mathsf{MVR}$, and $\delta_i^{(+)}(\$) = \mathsf{Accept}$, then $M_i^{(+)}$ accepts all words from ${\Sigma_1^{(i)}}^* \cdot \Sigma_3^{(i)} \cdot \Sigma^* \cup {\Sigma_1^{(i)}}^*$. Accordingly, we define $M_i'^{(+)}$ and $M_i''^{(+)}$ as above, but we define a third component $\hat{M}_i^{(+)}$ as follows:

$$\begin{aligned} &\hat{\delta}_i^{(+)}(¢) = \mathsf{MVR}, \\ &\hat{\delta}_i^{(+)}(a) = \mathsf{MVR} \quad \text{for all } a \in \Sigma_1^{(i)}, \\ &\hat{\delta}_i^{(+)}(\$) = \mathsf{Accept}. \end{aligned}$$

Then, in each successor set we replace $M_i^{(+)}$ by both, $M_i'^{(+)}$ and $\hat{M}_i^{(+)}$, and take $M_i''^{(+)}$ as the only successor of $M_i'^{(+)}$. Then together these three components accept the same words as $M_i^{(+)}$, but an accept instruction is only executed on the \$-symbol.

Finally we again split each component automaton $M_i'^{(+)}$ that contains more than one rewrite instruction into several automata, one for each letter that is deleted by a rewrite instruction. Then, the resulting stl-det-local-CD-R(1)-system is in normal form, and in mode = 1 it accepts the same language as the original system $\mathcal{M}$. □

Actually by again splitting the components $M_i'^{(+)}$, $M_i''^{(+)}$, and $\hat{M}_i^{(+)}$ into corresponding subcomponents, we can even obtain a stl-det-local-CD-R(1)-system that reduces each word to the empty word, and that only has a single accepting component M_+ that only accepts the empty word, that is, the transition function of M_+ is defined by $\delta_+(¢) = \mathsf{MVR}$ and $\delta_+(\$) = \mathsf{Accept}$.

We have seen that the language class $\mathcal{L}_{=1}$(stl-det-local-CD-R(1)) contains all regular languages and some languages that are not even context-free. Our next result implies that all languages from this class are semi-linear, that is, if $L \subseteq \Sigma^*$ belongs to this language class, and if $|\Sigma| = n$, then the Parikh image $\psi(L)$ of L is a semi-linear subset of $\mathbb{N}^n$.

Theorem 2. *Each language $L \in \mathcal{L}_{=1}$(stl-det-local-CD-R(1)) contains a regular sublanguage E such that $\psi(L) = \psi(E)$ holds. In fact, a finite-state acceptor for E can be constructed effectively from a* stl-det-local-CD-R(1)*-system for L.*

Proof. Let $\mathcal{M} = ((M_i, \sigma_i)_{i \in I}, I_0)$ be a stl-det-local-CD-R(1)-system over Σ, and let $L = L_{=1}(\mathcal{M})$. By Lemma 4 we can assume that $\mathcal{M}$ is in normal form. From $\mathcal{M}$ we construct a nondeterministic finite-state acceptor (NFA) A over Σ such that the language $L(A)$ is letter-equivalent to L.

For each index $i \in I$, let $M_i = (\Sigma, \Sigma, ¢, \$, 1, \delta_i)$, and let $\Sigma = \Sigma_1^{(i)} \cup \Sigma_2^{(i)} \cup \Sigma_3^{(i)} \cup \Sigma_4^{(i)}$ be the partitioning of Σ associated with M_i (see Definition 1). As $\mathcal{M}$ is in normal form, we see that $\Sigma_3^{(i)} = \emptyset$ and $|\Sigma_2^{(i)}| \leq 1$ for each $i \in I$. Further, we know that $\delta_i(¢) = \mathsf{MVR}$, and $\delta_i(\$) = \mathsf{Accept}$ if and only if $\Sigma_2^{(i)} = \emptyset$.

We now define the announced NFA $A = (Q, \Sigma, q_0, F, \delta_A)$ as follows:

- The set of states Q and the set of final states F are defined by

$$Q = I \cup \{q_0\} \cup \{\, q_\Delta \mid \Delta \subseteq \Sigma \,\} \text{ and } F = \{\, q_\Delta \mid \Delta \subseteq \Sigma \,\},$$

  that is, for each component automaton M_i, A has a particular state i, it has initial state q_0, and for each subalphabet Δ of Σ, it has an accepting state q_Δ.
- The transition relation δ_A is defined by:

$$\begin{aligned}
&(1)\ \delta_A(q_0, \varepsilon) &&= I_0, \\
&(2)\ \delta_A(i, a) &&= \sigma_i \qquad \text{for all } i \in I \text{ such that } a \in \Sigma_2^{(i)}, \\
&(3)\ \delta_A(i, \varepsilon) &&= \{q_{\Sigma_1^{(i)}}\} \text{ for all } i \in I \text{ such that } \delta_i(\$) = \mathsf{Accept}, \\
&(4)\ \delta_A(q_\Delta, a) &&= \{q_\Delta\} \quad \text{for all } \Delta \subseteq \Sigma \text{ and } a \in \Delta.
\end{aligned}$$

Then A is an NFA with ε-transitions that is easily constructed from $\mathcal{M}$. Hence, $L(A)$ is a regular language over Σ. It remains to prove that $L(A)$ is a

sublanguage of the language $L = L_{=1}(\mathcal{M})$ that is letter-equivalent to L. We first establish the following related technical result.

Claim 1. If $w = w_0 \vdash^c_{M_{i_1}} w_1 \vdash^c_{M_{i_2}} \cdots \vdash^c_{M_{i_s}} w_s \vdash^*_{M_{i_{s+1}}} \mathsf{Accept}$ is a mode $= 1$ computation of $\mathcal{M}$, then there exists a word $z \in \Sigma^*$ such that $i_1 z \vdash^*_A q \in F$ holds, and $\psi(z) = \psi(w)$.

Proof. We proceed by induction on the number s of cycles in the above computation. If $s = 0$, then $w = w_s$ is accepted by M_{i_1} through a tail computation. Thus, $w \in {\Sigma_1^{(i_1)}}^*$ and $\delta_{i_1}(\$) = \mathsf{Accept}$. Hence, A can perform the following computation:

$$i_1 w \vdash^{(3)}_A q_{\Sigma_1^{(i_1)}} w \vdash^{(4)^*}_A q_{\Sigma_1^{(i_1)}} \in F.$$

Thus, A accepts starting from $i_1 w$.

If $w = xay \vdash^c_{M_{i_1}} xy$, then $x \in {\Sigma_1^{(i_1)}}^*$, $a \in \Sigma_2^{(i_1)}$, and $i_2 \in \sigma_{i_1}$. Thus, A can perform the following step:

$$i_1 axy \vdash^{(2)}_A i_2 xy.$$

From the induction hypothesis we see that there exists a word $z_1 \in \Sigma^*$ that is accepted by A starting from the configuation $i_2 z_1$, and that is letter-equivalent to $w_1 = xy$. Hence, the word $z = az_1$ is accepted by A starting from the configuration $i_1 a z_1$, and az_1 is letter-equivalent to axy and therewith to $w = xay$. This completes the proof of Claim 1. □

If $w \in L_{=1}(\mathcal{M})$, then there exists an accepting mode $= 1$ computation of $\mathcal{M}$ of the following form:

$$w = w_0 \vdash^c_{M_{i_1}} w_1 \vdash^c_{M_{i_2}} \cdots \vdash^c_{M_{i_s}} w_s \vdash^*_{M_{i_{s+1}}} \mathsf{Accept}.$$

Then $i_1 \in I_0$, and from Claim 1 we see that there exists a word $z \in \Sigma^*$ such that z is letter-equivalent to w, and A accepts starting from the configuration $i_1 z$. But then $i_1 \in \delta_A(q_0, \varepsilon)$ implies that A accepts starting from the initial configuration $q_0 z$. Thus, $z \in L(A)$, that is, for each word $w \in L_{=1}(\mathcal{M})$, there exists a word $z \in L(A)$ such that z and w are letter-equivalent.

The proof of Theorem 2 is now completed by establishing the following claim.

Claim 2. If $z \in \Sigma^*$ and $i \in I$ such that A accepts starting from the configuration iz, then $\mathcal{M}$ has an accepting mode $= 1$ computation in which component automaton M_i starts from the initial tape contents $\¢z\$$.

Proof. We proceed by induction on the number of steps of group (2) that are applied in the accepting computation of A.

If no such step is applied at all, then the accepting computation of A has the following form:

$$iz \vdash^{(3)}_A q_{\Sigma_1^{(i)}} z \vdash^{(4)^*}_A q_{\Sigma_1^{(i)}} \in F.$$

From the definition of A we see that $\delta_i(\$) = \mathsf{Accept}$, and hence, component automaton M_i will accept starting from the tape contents $\¢z\$$.

Now assume that the accepting computation of A looks as follows:

$$iz = iav \vdash_A^{(2)} jv \vdash_A^* q_\Delta,$$

where $a \in \Sigma$, and $\Delta \subseteq \Sigma$. From the definition of A we see that $\delta_i(a) = \varepsilon$, and that $j \in \sigma_i$. Further, from the induction hypothesis we know that $\mathcal{M}$ has an accepting mode $= 1$ computation in which M_j starts from the tape contents $\mathbb{c}v\$$. It follows that there exists an accepting mode $= 1$ computation of $\mathcal{M}$ in which M_i starts with tape contents $\mathbb{c}av\$ = \mathbb{c}z\$$. □

It follows that each word $z \in L(A)$ belongs to the language $L_{=1}(\mathcal{M})$. Thus, $L(A)$ is indeed a regular sublanguage of L that is letter-equivalent to L. □

As all regular languages have semi-linear Parikh image, this yields the following important result.

Corollary 6. *The language class* $\mathcal{L}_{=1}$(stl-det-local-CD-R(1)) *only contains semi-linear languages, that is, if a language* L *over* $\Sigma = \{a_1, \ldots, a_n\}$ *is accepted by a CD-system of stateless deterministic* R*-automata of window size 1, then its Parikh image* $\psi(L)$ *is a semi-linear subset of* $\mathbb{N}^n$.

As the deterministic linear language $L = \{\, a^n b^n \mid n \geq 0 \,\}$ does not contain a regular sublanguage that is letter-equivalent to the language itself, we obtain the following non-inclusion result.

Proposition 6. *The language* $L = \{\, a^n b^n \mid n \geq 0 \,\}$ *is not accepted by any* stl-det-local-CD-R(1)*-system working in mode* $= 1$.

It follows analogously that the language $L_3 = \{\, a^n b^n c^n \mid n \geq 0 \,\}$ is not accepted by any stl-det-local-CD-R(1)-system working in mode $= 1$. As $L_3 = L_{abc} \cap a^* \cdot b^* \cdot c^*$, this implies the following in combination with Proposition 4.

Corollary 7. *The language class* $\mathcal{L}_{=1}$(stl-det-local-CD-R(1)) *is not closed under intersection with regular languages.*

Corollary 8. *The language class* $\mathcal{L}_{=1}$(stl-det-local-CD-R(1)) *is incomparable to the classes* DLIN*,* LIN*,* DCFL*, and* CFL *with respect to inclusion.*

Lemma 4 suggests to describe CD-systems of stateless deterministic R-automata of window size 1 by a graphical representation.

Let $\mathcal{M} = ((M_i, \sigma_i)_{i \in I}, I_0)$ be a stl-det-local-CD-R(1)-system in normal form on alphabet Σ, and for each $i \in I$, let $\Sigma = \Sigma_1^{(i)} \cup \Sigma_2^{(i)} \cup \Sigma_3^{(i)} \cup \Sigma_4^{(i)}$ be the partitioning of Σ associated with M_i (see Definition 1). Then we can describe $\mathcal{M}$ by a diagram that contains a vertex for each component automaton M_i and a special vertex "Accept". For all $i \in I$, if $\delta_i(\$) = \mathsf{Accept}$, then M_i accepts all words from ${\Sigma_1^{(i)}}^*$, and accordingly, there only is an edge labelled $\mathbb{c} \cdot {\Sigma_1^{(i)}}^* \cdot \$$ from vertex i to vertex "Accept" (see Figure 2). On the other hand, if $\delta_i(a) = \varepsilon$, then M_i deletes the leftmost occurrence of the letter a, provided it is preceded only

by a word from $\Sigma_1^{(i)^*}$. Accordingly, there is an edge labelled $(¢ \cdot \Sigma_1^{(i)^*}, a)$ from vertex i to vertex j for all $j \in \sigma_i$ (see Figure 3). Finally, vertex i is specifically marked for all initial indices $i \in I_0$. We illustrate this way of describing stl-det-local-CD-R(1)-systems by an example.

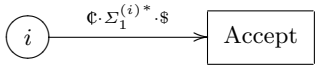

Fig. 2. A stl-det-R(1)-automaton M_i satisfying $\delta_i(\$) = \mathsf{Accept}$ accepts all words over $\Sigma_1^{(i)}$.

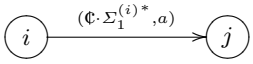

Fig. 3. A stl-det-R(1)-automaton M satisfying $\delta_i(a) = \varepsilon$ deletes the leftmost occurrence of the letter a, provided it is only preceded by a word over $\Sigma_1^{(i)}$. Further, it has an edge to vertex j for all $j \in \sigma_i$.

Example 3. Let $\mathcal{M} = ((M_i, \sigma_i)_{i \in I}, I_0)$ be the following system, where $I = \{1, 1', 2, 3, 4, +\}$, $I_0 = \{1\}$, $\sigma_1 = \{1', 2\}$, $\sigma_{1'} = \{1, 2\}$, $\sigma_2 = \{3\}$, $\sigma_3 = \{4\}$, $\sigma_4 = \{2, +\}$, $\sigma_+ = \{4\}$, and the various R-automata are given by the following transition functions:

$$
\begin{array}{lll}
M_1 : \delta_1(¢) = \mathsf{MVR}, & M_{1'} : \delta_{1'}(¢) = \mathsf{MVR}, & M_+ : \delta_+(¢) = \mathsf{MVR}, \\
\quad \delta_1(a) = \varepsilon, & \quad \delta_{1'}(a) = \varepsilon, & \quad \delta_+(\$) = \mathsf{Accept}, \\
M_2 : \delta_2(¢) = \mathsf{MVR}, & M_3 : \delta_3(¢) = \mathsf{MVR}, & M_4 : \delta_4(¢) = \mathsf{MVR}, \\
\quad \delta_2(a) = \varepsilon, & \quad \delta_3(a) = \mathsf{MVR}, & \quad \delta_4(a) = \mathsf{MVR}, \\
\quad \delta_2(b) = \mathsf{MVR}, & \quad \delta_3(b) = \varepsilon, & \quad \delta_4(b) = \mathsf{MVR}, \\
\quad \delta_2(c) = \mathsf{MVR}, & \quad \delta_3(c) = \mathsf{MVR}, & \quad \delta_4(c) = \varepsilon.
\end{array}
$$

Then using the component automata M_1 and M_1', $\mathcal{M}$ deletes a positive number of a's, and then using component automata M_2, M_3, and M_4 it deletes an equal number of a's, b's, and c's, before it accepts the empty word by component automaton M_+. Thus,

$$L_{=1}(\mathcal{M}) = \{\, a^n w \mid n \geq 1,\ w \in \{a, b, c\}^+ \text{ satisfying } |w|_a = |w|_b = |w|_c \,\}.$$

Now this CD-system of stateless R-automata of window size 1 can be described more compactly by the diagram given in Figure 4.

As another example, we consider the language

$$L_2 = \{\, w a^n \mid n \geq 1,\ w \in \{a, b, c\}^+ \text{ satisfying } |w|_a = |w|_b = |w|_c \,\}.$$

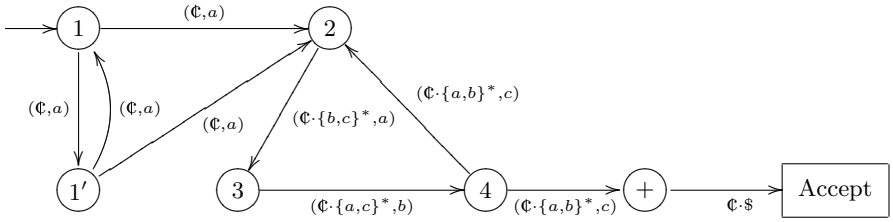

Fig. 4. The stl-det-CD-R(1)-system $\mathcal{M}$ from Example 3.

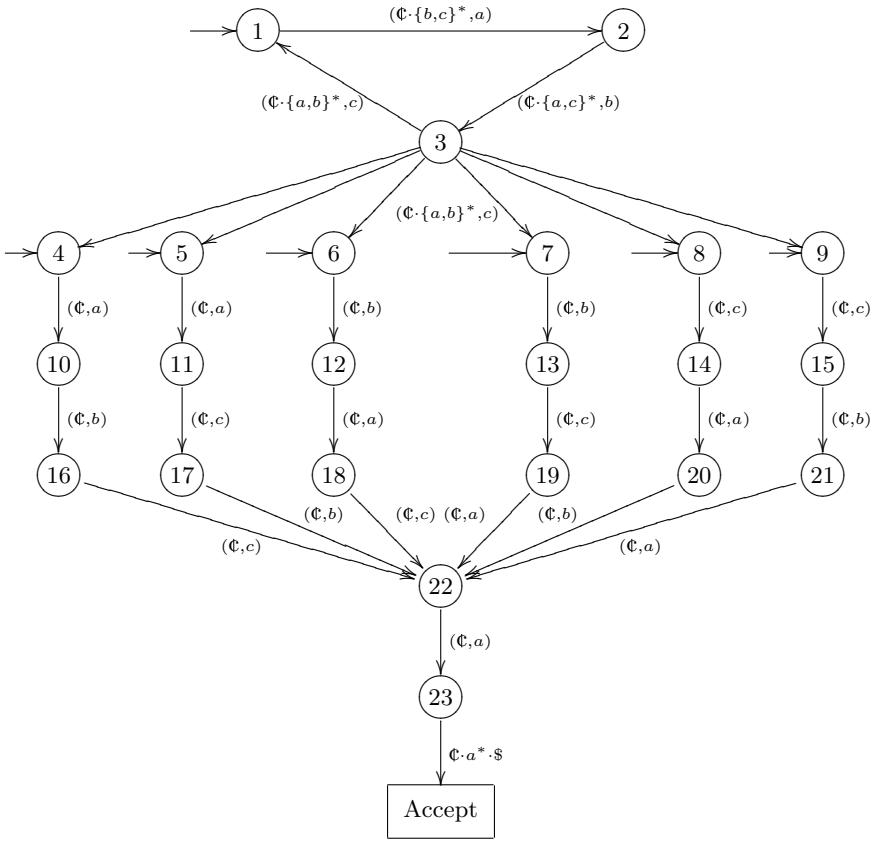

Fig. 5. The stl-det-CD-R(1)-system $\mathcal{M}$ for the language L_2.

Example 4. Let $\mathcal{M}$ be the CD-system of stateless deterministic R-automata of window size 1 that is described by the diagram in Figure 5.

The system $\mathcal{M}$ consists of 23 component automata, 7 of which are initial automata. Automaton M_{23} is the only one with an accept instruction. It accepts the language a^*. Accordingly, computations that begin with the initial automaton M_4 accept the regular language $abc \cdot a^+$, those that begin with M_5 accept the language $acb \cdot a^+$, and analogously for those computations that begin with M_6, M_7, M_8 or M_9. It follows that in combination these computations accept the language

$$L_2' = \{\, wa^n \mid n \geq 1,\ w \in \{a,b,c\}^+ \text{ satisfying } |w|_a = |w|_b = |w|_c = 1 \,\}.$$

An accepting computation that begins with the initial automaton M_1 consists of two parts: first it cycles through the automata M_1, M_2, and M_3, in each round deleting the first a, b, and c from the left, and then it continues with a computation that accepts a word $w_1 \cdot a^n$ from L_2', where $|w_1|_a = |w_1|_b = |w_1|_c = 1$. Since at that moment there is at most a single a to the left of the last remaining letters b and c, it follows that all deletions in the first phase of this computation where executed to the left of the suffix a^n. Hence, the input w does indeed belong to the language L_2, which implies that $L_{=1}(\mathcal{M}) = L_2$ holds.

5 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 dependency relation D (or rather its associated 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 equivalence 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., [7]). By φ_D we denote the morphism $\varphi_D : \Sigma^* \to M(D)$ that is defined by $w \mapsto [w]_D$ for all words $w \in \Sigma^*$.

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., [7] Claim A in the proof of Prop. 6.2.2) will be useful in what follows.

Proposition 7. *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 $S = \alpha^{-1}(P)$ [1]. Accordingly, this property can be characterized as follows (see [7] Prop. 6.1.10).

Proposition 8. *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 [1]. This property can be characterized more conveniently as follows.

Proposition 9. *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., [7]).

Proposition 10. *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 9 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. The next theorem states that all these languages are accepted by stl-det-local-CD-R(1)-systems.

Theorem 3. *Let $M(D)$ be the trace monoid presented by (Σ, D), where D is a dependency relation on the finite alphabet Σ. Then*

$$\mathcal{LRAT}(D) \subseteq \mathcal{L}_{=1}(\mathsf{stl\text{-}det\text{-}local\text{-}CD\text{-}R(1)}),$$

that is, the language $\varphi_D^{-1}(S)$ is accepted by a stl-det-local-CD-R(1)*-system working in mode $= 1$ for each rational set of traces $S \subseteq M(D)$.*

Proof. Let S be a rational subset of $M(D)$. Then there exists a regular language R over Σ such that $S = \varphi_D(R)$. As $R \subseteq \Sigma^*$ is a regular language, there exists a complete deterministic finite-state acceptor $A = (Q, \Sigma, p_0, F, \delta)$ for R. From A we now construct a stl-det-local-CD-R(1)-system $\mathcal{M} = ((M_i, \sigma_i)_{i \in I}, I_0)$ as follows (cf. the proof of Proposition 5):

- The set of indices is $I = (Q \times \Sigma) \cup (Q' \times \Sigma) \cup \{+\}$, where $Q' = \{\, q' \mid q \in Q \,\}$ is a copy of Q such that $Q \cap Q' = \emptyset$,
- the set of initial indices is $I_0 = \begin{cases} \{\, (p_0, a) \mid a \in \Sigma \,\}, & \text{if } \varepsilon \notin L, \\ \{\, (p_0, a) \mid a \in \Sigma \,\} \cup \{+\}, & \text{if } \varepsilon \in L, \end{cases}$
- the successor relations are defined by
  - $\sigma_{(q,a)} = \begin{cases} \{\, (\delta(q,a), b) \mid b \in \Sigma \,\} \cup \{+\}, & \text{if } \delta(q,a) \neq q \text{ and } \delta(q,a) \in F, \\ \{\, (\delta(q,a), b) \mid b \in \Sigma \,\}, & \text{if } \delta(q,a) \neq q \text{ and } \delta(q,a) \notin F, \\ \{\, (q', b) \mid b \in \Sigma \,\} \cup \{+\}, & \text{if } \delta(q,a) = q \text{ and } q \in F, \\ \{\, (q', b) \mid b \in \Sigma \,\}, & \text{if } \delta(q,a) = q \text{ and } q \notin F, \end{cases}$
  - $\sigma_{(q',a)} = \begin{cases} \{\, (\delta(q,a), b) \mid b \in \Sigma \,\} \cup \{+\}, & \text{if } \delta(q,a) \in F, \\ \{\, (\delta(q,a), b) \mid b \in \Sigma \,\}, & \text{if } \delta(q,a) \notin F, \end{cases}$
  - $\sigma_+ = \{\, (p_0, a) \mid a \in \Sigma \,\}$,
- and the stl-det-R(1)-automata $M_{(q,a)}$, $M_{(q',a)}$, and M_+ are defined by the following transition functions:

$$\begin{array}{lrcl} M_{(q,a)} : & \delta_{(q,a)}(\mathfrak{c}) & = & \mathsf{MVR}, \\ & \delta_{(q,a)}(b) & = & \mathsf{MVR} \quad \text{for all } b \in \Sigma \text{ satisfying } (b,a) \in I_D, \\ & \delta_{(q,a)}(a) & = & \varepsilon, \\ M_{(q',a)} : & \delta_{(q',a)}(\mathfrak{c}) & = & \mathsf{MVR}, \\ & \delta_{(q',a)}(b) & = & \mathsf{MVR} \quad \text{for all } b \in \Sigma \text{ satisfying } (b,a) \in I_D, \\ & \delta_{(q',a)}(a) & = & \varepsilon, \\ M_+ : & \delta_+(\mathfrak{c}) & = & \mathsf{MVR}, \\ & \delta_+(\$) & = & \mathsf{Accept}. \end{array}$$

It remains to show that $L_{=1}(\mathcal{M}) = \varphi_D^{-1}(S) = \bigcup_{u \in R}[u]_D$.

Claim 1. $\bigcup_{u \in R}[u]_D \subseteq L_{=1}(\mathcal{M})$.

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 $i = 1, \ldots, n$, 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 $w_i \in L_{=1}(\mathcal{M})$ for all i by induction on i.

For $i = 0$ we have $w_0 = u \in R$. Thus, w_0 is accepted by the finite-state acceptor A, and it follows from the proof of Proposition 5 that w_0 is also accepted by a mode $= 1$ computation of $\mathcal{M}$.

Now assume that $w_i \in L_{=1}(\mathcal{M})$ 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 $\mathcal{M}$ has an accepting mode $= 1$ computation for $w_i = xaby$, which is of one of the following two forms:

$$w_i = xaby \vdash_{\mathcal{M}}^{c^m} x'aby' \vdash_{M_{(q,a)}}^{c} x'by' \vdash_{\mathcal{M}}^{c^*} \varepsilon \vdash_{M_+}^{*} \mathsf{Accept},$$

or

$$w_i = xaby \vdash_{\mathcal{M}}^{c^m} x'aby' \vdash_{M_{(q,b)}}^{c} x'ay' \vdash_{\mathcal{M}}^{c^*} \varepsilon \vdash_{M_+}^{*} \mathsf{Accept},$$

where in the first m cycles some letters from x and y are deleted, in this way reducing these factors to x' and y', respectively, and $q \in Q \cup Q'$ is a state (or a copy of a state) of A. However, as $(a,b) \in I$, the component automaton $M_{(q,a)}$ (or $M_{(q,b)}$) can read across the letter b (or a) when looking for the leftmost occurrence of the letter a (or b). Thus, $\mathcal{M}$ also has an accepting mode $= 1$ computation for $w_{i+1} = xbay$, which is of one of the following two forms:

$$w_{i+1} = xbay \vdash_{\mathcal{M}}^{c^m} x'bay' \vdash_{M_{(q,a)}}^{c} x'by' \vdash_{\mathcal{M}}^{c^*} \varepsilon \vdash_{M_+}^{*} \mathsf{Accept},$$

or

$$w_{i+1} = xbay \vdash_{\mathcal{M}}^{c^m} x'bay' \vdash_{M_{(q,b)}}^{c} x'ay' \vdash_{\mathcal{M}}^{c^*} \varepsilon \vdash_{M_+}^{*} \mathsf{Accept},$$

implying that $w_{i+1} \in L_{=1}(\mathcal{M})$. This completes the proof of Claim 1. □

Claim 2. $L_{=1}(\mathcal{M}) \subseteq \bigcup_{u \in R} [u]_D$.

Proof. Let $w \in L_{=1}(\mathcal{M})$, and let

$$\begin{array}{cccccc} w = w_n & \vdash_{M_{(q_n,a_n)}}^{c} & w_{n-1} & \vdash_{M_{(q_{n-1},a_{n-1})}}^{c} & w_{n-2} & \vdash^{c} \\ \cdots & \vdash_{M_{(q_2,a_2)}}^{c} & w_1 & \vdash_{M_{(q_1,a_1)}}^{c} & w_0 = \varepsilon & \vdash_{M_+}^{*} \mathsf{Accept} \end{array}$$

be an accepting mode $= 1$ computation of $\mathcal{M}$ on input w, where $q_n, q_{n-1}, \ldots, q_1$ are states of A (or copies thereof) and $(q_n, a_n) \in I_0$. We claim that, for each $i = 1, \ldots, n$, there exists a word $u_i \in \Sigma^*$ such that $u_i \equiv_D w_i$ and $\delta(q_i, u_i) \in F$, that is, the finite-state acceptor 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 $M_+ \in \sigma_{M_{(q_1,a_1)}}$. From the definition of $\mathcal{M}$ we conclude that $\delta(q_1, a_1) \in F$, that is, 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 $\delta(q_i, u_i) \in F$ hold. The above computation of $\mathcal{M}$ contains the cycle $w_{i+1} \vdash_{M_{(q_{i+1},a_{i+1})}}^{c} w_i$, and $(q_i, a_i) \in \sigma_{(q_{i+1},a_{i+1})}$. Again from the definition of $\mathcal{M}$ we see that $\delta(q_{i+1}, a_{i+1}) = q_i$, and that $w_{i+1} = xa_{i+1}y$ and $w_i = xy$ for some words $x, y \in \Sigma^*$ such that $(x, a_{i+1}) \in I_D$. Let u_{i+1} be the word $u_{i+1} = a_{i+1}u_i$. 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 $\delta(q_{i+1}, u_{i+1}) = \delta(q_{i+1}, a_{i+1}u_i) = \delta(\delta(q_{i+1}, a_{i+1}), u_i) = \delta(q_i, u_i) \in F$.

For $i = n$ we obtain a word $u \in \Sigma^*$ such that $u \equiv_D w$, and A accepts u starting from state $q_n = p_0$. Hence, $u \in R$, and it follows that $L_{=1}(\mathcal{M}) \subseteq \bigcup_{u \in R}[u]_D$ holds. □

Now Claims 1 and 2 together show that $L_{=1}(\mathcal{M}) = \bigcup_{u \in R}[u]_D = \varphi_D^{-1}(S)$, which completes the proof of Theorem 3. □

Observe that the Dyck language $D_1'^*$ is not a rational trace language. Thus, the language class $\mathcal{L}_{=1}(\mathsf{stl\text{-}det\text{-}local\ CD\text{-}R(1)})$ is a proper superclass of the class of all rational trace languages.

Next we present a restricted class of stl-det-local-CD-R(1)-systems that accept exactly the rational trace languages by mode = 1 computations.

Definition 3. *Let $\mathcal{M} = ((M_i, \sigma_i)_{i \in I}, I_0)$ be a* stl-det-local-CD-R(1)*-system in normal form on Σ that satisfies the following condition:*

$$(*) \qquad \forall i, j \in I : \Sigma_2^{(i)} = \Sigma_2^{(j)} \text{ implies that } \Sigma_1^{(i)} = \Sigma_1^{(j)},$$

that is, if two component automata erase the same letter, then they also read across the same subset of Σ. With $\mathcal{M}$ we associate a binary relation

$$I_{\mathcal{M}} = \bigcup_{i \in I}(\Sigma_1^{(i)} \times \Sigma_2^{(i)}),$$

that is, $(a, b) \in I_{\mathcal{M}}$ if and only if there exists a component automaton M_i such that $\delta_i(a) =$ MVR and $\delta_i(b) = \varepsilon$. Further, by $D_{\mathcal{M}}$ we denote the relation $D_{\mathcal{M}} = (\Sigma \times \Sigma) \setminus I_{\mathcal{M}}$.

Observe that the relation $I_{\mathcal{M}}$ defined above is necessarily irreflexive, but that it will in general not be symmetric. For example, consider the system $\mathcal{M}$ from the proof of Proposition 3. It is in normal form, but the corresponding relation $I_{\mathcal{M}} = \{(a, b)\}$ is not symmetric. And indeed, the language $L_{=1}(\mathcal{M})$ is the Dyck language $D_1'^*$, which is not a rational trace language.

Theorem 4. *Let $\mathcal{M}$ be a* stl-det-local-CD-R(1)*-system over Σ satisfying condition $(*)$ above. If the associated relation $I_{\mathcal{M}}$ is symmetric, then $L_{=1}(\mathcal{M})$ is a rational trace language over Σ. In fact, from $\mathcal{M}$ one can construct a finite-state acceptor B over Σ such that $L_{=1}(\mathcal{M}) = \varphi_{D_{\mathcal{M}}}^{-1}(\varphi_{D_{\mathcal{M}}}(L(B)))$.*

Proof. Let $\mathcal{M} = ((M_i, \sigma_i)_{i \in I}, I_0)$ be a stl-det-local-CD-R(1)-system in normal form on Σ that satisfies condition $(*)$, and assume that the associated relation $I_{\mathcal{M}} = \bigcup_{i \in I}(\Sigma_1^{(i)} \times \Sigma_2^{(i)})$ is symmetric. Then the relation $D_{\mathcal{M}} = (\Sigma \times \Sigma) \setminus I_{\mathcal{M}}$ is reflexive and symmetric, and so it is a dependency relation on Σ with associated independence relation $I_{\mathcal{M}}$. Without loss of generality we may assume that all letters from Σ do actually occur in some words of $L_{=1}(\mathcal{M})$, since otherwise we could simply remove these letters from Σ. In addition, we can assume that $\mathcal{M}$ has only a single accepting component automaton M_+, and that M_+ only accepts the empty word. From the properties of $\mathcal{M}$ we obtain the following consequences:

1. As all words $w \in L_{=1}(\mathcal{M})$ are first reduced to the empty word, which is then accepted by the accepting component automaton of $\mathcal{M}$, we see that, for each letter $a \in \Sigma$, there exists a component automaton M_i such that $\Sigma_2^{(i)} = \{a\}$.
2. If $(a,b) \in I_{\mathcal{M}}$, then $a \in \Sigma_1^{(i)}$ for all component automata M_i for which $\Sigma_2^{(i)} = \{b\}$ holds.
3. If $(a,b) \in I_{\mathcal{M}}$, then $(b,a) \in I_{\mathcal{M}}$, too, and hence, $b \in \Sigma_1^{(j)}$ for all component automata M_j for which $\Sigma_2^{(j)} = \{a\}$ holds.

Let $L = L_{=1}(\mathcal{M})$. We claim that L is a rational trace language over the trace monoid defined by $(\Sigma, D_{\mathcal{M}})$, that is, $L \in \mathcal{LRAT}(D_{\mathcal{M}})$. To verify this claim we present a regular language $R \subseteq \Sigma^*$ such that $L = \bigcup_{u \in R}[u]_{D_{\mathcal{M}}}$.

The regular language R will be defined through a nondeterministic finite-state acceptor (with ε-moves) $B = (Q, \Sigma, p_0, p_+, \delta)$, where Q is a finite set of states, $p_0 \in Q$ is the initial state, $p_+ \in Q$ is the only final state, and $\delta \subseteq (Q \times (\Sigma \cup \{\varepsilon\}) \times Q)$ is the transition relation. This finite-state acceptor is obtained from $\mathcal{M}$ as follows. Here $I_r = I \setminus \{+\}$ is the subset of I containing all component automata that perform a rewrite operation, $i \in I_r$, and $a \in \Sigma$:

$$\begin{array}{lll}
Q & = \{p_0, p_+\} \cup \{\, q_i \mid i \in I_r \,\}, & \\
\delta(p_0, \varepsilon) & = \{\, q_i \mid i \in I_0 \,\}, & \text{if } + \notin I_0, \\
\delta(p_0, \varepsilon) & = \{\, q_i \mid i \in I_0 \cap I_r \,\} \cup \{p_+\}, & \text{if } + \in I_0, \\
\delta(q_i, a) & = \{\, q_j \mid j \in \sigma_i \,\}, & \text{if } \{a\} = \Sigma_2^{(i)} \text{ and } + \notin \sigma_i, \\
\delta(q_i, a) & = \{\, q_j \mid j \in \sigma_i \cap I_r \,\} \cup \{p_+\}, & \text{if } \{a\} = \Sigma_2^{(i)} \text{ and } + \in \sigma_i, \\
\delta(q, a) & = \emptyset & \text{for all other cases.}
\end{array}$$

Now $R = L(B)$ is the announced regular language over Σ. It remains to prove that $L = \bigcup_{u \in R}[u]_{D_{\mathcal{M}}}$ holds.

Claim 1. $\bigcup_{u \in R}[u]_{D_{\mathcal{M}}} \subseteq L$.

Proof. First we show that $R \subseteq L$ holds. Indeed if we remove all MVR-operations that read across letters of Σ from all the rewriting component automata of $\mathcal{M}$, then we obtain a stl-det-local-CD-R(1)-system $\mathcal{M}'$ that deletes a word letter by letter from the left to the right. Now the finite-state acceptor B simply simulates the system $\mathcal{M}'$, which implies that

$$R = L(B) = L_{=1}(\mathcal{M}') \subseteq L_{=1}(\mathcal{M}) = L$$

holds.

Let $w \equiv_{D_{\mathcal{M}}} u \in R$, and let $u = w_0, w_1, \ldots, w_n = w$ be a sequence of words such that, for each $i = 1, \ldots, n$, w_i is obtained from w_{i-1} by replacing a factor ab by ba for some pair of letters $(a,b) \in I_{\mathcal{M}}$. 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 $w_0 \in L$ by the considerations in the previous paragraph. 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 $(a,b) \in I_{\mathcal{M}}$. By our hypothesis $\mathcal{M}$ has an

accepting mode = 1 computation for $w_i = xaby$, which is of one of the following two forms:

$$w_i = xaby \vdash_{\mathcal{M}}^{c^k} x_1aby_1 \vdash_{M_i}^{c} x_1by_1 \vdash_{\mathcal{M}}^{c^l} x_2by_2 \vdash_{M_j}^{c} x_2y_2 \vdash_{\mathcal{M}}^{*} \mathsf{Accept},$$

or

$$w_i = xaby \vdash_{\mathcal{M}}^{c^k} x_1aby_1 \vdash_{M_{j'}}^{c} x_1ay_1 \vdash_{\mathcal{M}}^{c^l} x_2ay_2 \vdash_{M_{i'}}^{c} x_2y_2 \vdash_{\mathcal{M}}^{*} \mathsf{Accept},$$

where in the first k cycles some letters from x and y are deleted, in this way reducing these factors to x_1 and y_1, respectively, $\Sigma_2^{(i)} = \{a\} = \Sigma_2^{(i')}$ and $\Sigma_2^{(j)} = \{b\} = \Sigma_2^{(j')}$, and in the latter l cycles some letters from x_1 and y_1 are deleted, reducing these factors to x_2 and y_2, respectively. As $(a, b) \in I_{\mathcal{M}}$, we see from the above stated properties of $\mathcal{M}$ that $b \in \Sigma_1^{(i)}$. Hence, in the former case we obtain the mode = 1 computation

$$w_{i+1} = xbay \vdash_{\mathcal{M}}^{c^k} x_1bay_1 \vdash_{M_i}^{c} x_1by_1 \vdash_{\mathcal{M}}^{c^l} x_2by_2 \vdash_{M_j}^{c} x_2y_2 \vdash_{\mathcal{M}}^{*} \mathsf{Accept},$$

while in the latter case we obtain the mode = 1 computation

$$w_{i+1} = xbay \vdash_{\mathcal{M}}^{c^k} x_1bay_1 \vdash_{M_{j'}}^{c} x_1ay_1 \vdash_{\mathcal{M}}^{c^l} x_2ay_2 \vdash_{M_{i'}}^{c} x_2y_2 \vdash_{\mathcal{M}}^{*} \mathsf{Accept}.$$

Thus, we see that $w = w_n$ is accepted by a mode = 1 computation of $\mathcal{M}$, which completes the proof of Claim 1. □

Claim 2. $L \subseteq \bigcup_{u \in R} [u]_{D_{\mathcal{M}}}$.

Proof. Let $w \in L$, and let

$$\begin{array}{cccccc} w = w_n & \vdash_{M_{i_n}}^{c} & w_{n-1} & \vdash_{M_{i_{n-1}}}^{c} & w_{n-2} & \vdash^{c} \\ \cdots & \vdash_{M_{i_2}}^{c} & w_1 & \vdash_{M_{i_1}}^{c} & w_0 = \varepsilon & \vdash_{M_+}^{*} \mathsf{Accept} \end{array}$$

be an accepting mode = 1 computation of $\mathcal{M}$ on input w. We claim that, for each $j = 1, \ldots, n$, there exists a word $u_j \in \Sigma^*$ such that $u_j \equiv_{D_{\mathcal{M}}} w_j$ and $p_+ \in \delta(q_{i_j}, u_j)$, that is, the finite-state acceptor B accepts the word u_j when starting from state q_{i_j}.

We prove this claim by induction on j. For $j = 1$ we have $w_j = a_1$, where $\Sigma_2^{(i_1)} = \{a_1\}$, and $+ \in \sigma_{i_1}$. From the definition of B we conclude that $p_+ \in \delta(q_{i_1}, a_1)$, that is, we can simply take $u_1 = a_1 = w_1$. Now assume that, for some $j \geq 1$, $u_j \equiv_{D_{\mathcal{M}}} w_j$ and $p_+ \in \delta(q_{i_j}, u_j)$ hold. The above computation of $\mathcal{M}$ contains the cycle $w_{j+1} \vdash_{M_{i_{j+1}}}^{c} w_j$, that is, $w_{j+1} = xa_{j+1}y$ and $w_j = xy$ for some words $x, y \in \Sigma^*$ and the letter a_{j+1} satisfying $\Sigma_2^{(i_{j+1})} = \{a_{j+1}\}$, and $i_j \in \sigma_{i_{j+1}}$. Also we see that $(x, a_{j+1}) \in I_{\mathcal{M}}$. Again from the definition of B it follows that $q_{i_j} \in \delta(q_{i_{j+1}}, a_{j+1})$. Now let u_{j+1} be the word $u_{j+1} = a_{j+1}u_j$. Then

$$u_{j+1} = a_{j+1}u_j \equiv_{D_{\mathcal{M}}} a_{j+1}w_j = a_{j+1}xy \equiv_{D_{\mathcal{M}}} xa_{j+1}y = w_{j+1},$$

and $\delta(q_{i_{j+1}}, u_{j+1}) = \delta(q_{i_{j+1}}, a_{j+1}u_j) = \delta(\delta(q_{i_{j+1}}, a_{j+1}), u_j) \supseteq \delta(q_{i_j}, u_j) \ni p_+$. Finally, for $j = n$ we obtain a word u such that $u \equiv_{D_\mathcal{M}} w$ and $p_+ \in \delta(p_0, u)$, which means that $u \in R$. Thus, it follows that $L \subseteq \bigcup_{u \in R}[u]_{D_\mathcal{M}}$ holds. □

Now Claims 1 and 2 together show that $L = L_{=1}(\mathcal{M}) = \bigcup_{u \in R}[u]_{D_\mathcal{M}}$, which completes the proof of Theorem 4. □

Observe that the system $\mathcal{M}$ constructed in the proof of Theorem 3 is in normal form, that it satisfies property $(*)$, and that the associated relation $I_\mathcal{M}$ coincides with the relation I_D, and hence, it is symmetric. Thus, Theorems 3 and 4 together yield the following characterization.

Corollary 9. *A language $L \subseteq \Sigma^*$ is a rational trace language if and only if there exists a* stl-det-local-CD-R(1)*-system $\mathcal{M}$ in accepting normal form satisfying condition $(*)$ such that the relation $I_\mathcal{M}$ is symmetric and $L = L_{=1}(\mathcal{M})$.*

In the proof of Theorem 3 we effectively constructed a stl-det-local-CD-R(1)-system for the rational trace language $\varphi_D^{-1}(\varphi_D(R))$ from a finite-state acceptor for the regular language R. Hence, if $S_1, S_2 \subseteq M(D)$ are rational subsets of the trace monoid $M(D)$, then we can construct finite-state acceptors B_1 and B_2 from stl-det-local-CD-R(1)-systems $\mathcal{M}_1$ for $L_1 = \varphi_D^{-1}(S_1)$ and $\mathcal{M}_2$ for $L_2 = \varphi_D^{-1}(S_2)$ such that $S_1 = \varphi_D(R_1)$ and $S_2 = \varphi_D(R_2)$, where $R_i = L(B_i)$, $i = 1, 2$. It is easily seen that $S_1 \cup S_2 = \varphi_D(R_1 \cup R_2)$, $S_1 \cdot S_2 = \varphi_D(R_1 \cdot R_2)$, and $S_1^* = \varphi_D(R_1^*)$. From B_1 and B_2 we can construct finite-state acceptors for the languages $R_1 \cup R_2$, $R_1 \cdot R_2$, and R_1^*. Thus, Theorem 4 shows that we can construct stl-det-local-CD-R(1)-systems for the languages $\varphi_D^{-1}(S_1 \cup S_2)$, $\varphi_D^{-1}(S_1 \cdot S_2)$, and $\varphi_D^{-1}(S_1^*)$. Hence, the stl-det-local-CD-R(1)-systems of Corollary 9 form an effective calculus for rational trace languages. However, a stl-det-local-CD-R(1)-system may accept a rational trace language, even if it does not satisfy all the additional restrictions above. Hence, the following problem remains.

Open Problem 4. *Is there a syntactic characterization for those* stl-det-local-CD-R(1)*-systems that accept rational trace languages by mode = 1 computations?*

6 Closure Properties

In Corollary 7 we have seen that the language class $\mathcal{L}_{=1}$(stl-det-local-CD-R(1)) is not closed under intersection with regular languages. Here we derive further non-closure properties, but also a number of closure properties for this class.

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,

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

If L is accepted by a stl-det-local-CD-R(1)-system $\mathcal{M}$, then from $\mathcal{M}$ we can construct a finite-state acceptor B for a regular sublanguage E of L that is

letter-equivalent to L (Theorem 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 $M(D)$ presented by (Σ, 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 3 that this language is accepted by a stl-det-local-CD-R(1)-system $\mathcal{M}'$. In fact, the system $\mathcal{M}'$ can effectively be constructed from the finite-state acceptor B, and therewith from the given stl-det-local-CD-R(1)-system $\mathcal{M}$. This yields the following effective closure property.

Corollary 10. *The language class* $\mathcal{L}_{=1}$(stl-det-local-CD-R(1)) *is effectively 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 3 implies the following result.

Corollary 11. *All commutative semi-linear languages are contained in the language class* $\mathcal{L}_{=1}$(stl-det-local-CD-R(1)).

Next we consider the closure under Boolean operations.

Proposition 11.

(a) *The language class* $\mathcal{L}_{=1}$(stl-det-local-CD-R(1)) *is closed under union.*
(b) *The language class* $\mathcal{L}_{=1}$(stl-det-local-CD-R(1)) *is neither closed under intersection nor under complementation.*

Proof. (a) Let $\mathcal{M} = ((M_i, \sigma_i)_{i \in I}, I_0)$ and $\mathcal{M}' = ((M'_i, \sigma'_i)_{i \in I'}, I'_0)$ be stl-det-local-CD-R(1)-systems with disjoint sets of indices I and I'. We define a new stl-det-local-CD-R(1)-system $\tilde{\mathcal{M}} = ((\tilde{M}_i, \tilde{\sigma}_i)_{i \in \tilde{I}}, \tilde{I}_0)$ by taking $\tilde{I} = I \cup I'$, $\tilde{I}_0 = I_0 \cup I'_0$,

$$\tilde{M}_i = \left\{ \begin{array}{l} M_i, \text{ for } i \in I \\ M'_i, \text{ for } i \in I' \end{array} \right\}, \text{ and } \tilde{\sigma}_i = \left\{ \begin{array}{l} \sigma_i, \text{ for } i \in I \\ \sigma'_i, \text{ for } i \in I' \end{array} \right\}.$$

Then $\tilde{\mathcal{M}}$ is the disjoint union of the two given systems, and it follows immediately that $L_{=1}(\tilde{\mathcal{M}}) = L_{=1}(\mathcal{M}) \cup L_{=1}(\mathcal{M}')$. This proves that the class $\mathcal{L}_{=1}$(stl-det-local-CD-R(1)) is closed under union.
(b) From Proposition 5 and Corollary 7 we see that this language class is not closed under intersection. Now closure under union and non-closure under intersection imply that this class is not closed under complementation, either. □

We now turn to the product operation. We will show that the language class $\mathcal{L}_{=1}$(stl-det-local-CD-R(1)) is closed under product, that is, if L_1 and L_2

are accepted by stl-det-local-CD-R(1)-systems, then so is the language $L_1 \cdot L_2 = \{ uv \mid u \in L_1, v \in L_2 \}$.

Obviously we can assume that the stl-det-local-CD-R(1)-system $\mathcal{M}_1$ accepting the language L_1 is in normal form. In fact, we can even assume that it only has a single accepting component M_+, and that this component only accepts the empty word. Thus, $\mathcal{M}_1$ reduces a given input word $w \in L_1$ first to the empty word by performing $|w|$ many cycles, and then it accepts by activating M_+. Now it would appear that we obtain a stl-det-local-CD-R(1)-system $\mathcal{M}$ for the language $L_1 \cdot L_2$ by simply replacing the component M_+ of $\mathcal{M}_1$ by the initial components of the system $\mathcal{M}_2$ for the language L_2. However, the situation is not that easy as shown by the following example.

Example 5. Consider the following language L_1 on $\Sigma = \{a, b, c, d\}$, where $D_1'^*$ denotes the Dyck language on $\{a, b\}$, $\hat{D}_1^*$ denotes the Dyck language on $\{c, d\}$, and sh denotes the *shuffle*:

$$L_1 = \{ w \in \Sigma^+ \mid w \in \mathrm{sh}(D_1'^*, \hat{D}_1^*) \text{ such that } \forall\, x, y, z : w = xcydz \wedge |x|_c = |xy|_d \text{ imply } |x|_a \geq |xy|_b \},$$

and let $\mathcal{M}_1 = ((M_i, \sigma_i)_{i \in I}, I_0)$ be the stl-det-local-CD-R(1)-system that is specified by $I = \{1, 2, 3, 4, +\}$, $I_0 = \{1, 3\}$, $\sigma_1 = \{2\}$, $\sigma_2 = \{1, 3, +\}$, $\sigma_3 = \{4\}$, $\sigma_4 = \{1, 3, +\}$, $\sigma_+ = \{1, 3\}$, and the R-automata $M_1, \ldots, M_4, M_+$ are defined through the following transition functions:

$$\begin{array}{lll}
M_1 : & (1)\ \delta_1(\mathrm{¢}) = \mathsf{MVR}, & \\
& (2)\ \delta_1(a) = \varepsilon, & \\
M_2 : & (3)\ \delta_2(\mathrm{¢}) = \mathsf{MVR}, & \\
& (4)\ \delta_2(x) = \mathsf{MVR} & \text{for all } x \in \{a, c, d\}, \\
& (5)\ \delta_2(b) = \varepsilon, & \\
M_3 : & (6)\ \delta_3(\mathrm{¢}) = \mathsf{MVR}, & \\
& (7)\ \delta_3(c) = \varepsilon, & \\
M_4 : & (8)\ \delta_4(\mathrm{¢}) = \mathsf{MVR}, & \\
& (9)\ \delta_4(x) = \mathsf{MVR} & \text{for all } x \in \{a, c\}, \\
& (10)\ \delta_4(d) = \varepsilon, & \\
M_+ : & (11)\ \delta_+(\mathrm{¢}) = \mathsf{MVR}, & \\
& (12)\ \delta_+(\$) = \mathsf{Accept}. &
\end{array}$$

Claim 1. $L_1 = L_{=1}(\mathcal{M}_1)$.

Proof. Let $w \in L_1$, $w \neq \varepsilon$, be given as input. As L_1 is contained in the shuffle product of $D_1'^*$ and $\hat{D}_1^*$, each element of L starts with an occurrence of a letter a or c. Accordingly, if $w = aubv$, where $|u|_b = 0$, then the $\mathcal{M}_1$-computation starts with component automaton M_1, that is, it starts by executing the cycles $w = aubv \vdash^{\mathrm{c}}_{M_1} ubv \vdash^{\mathrm{c}}_{M_2} uv$. On the other hand, if $w = cu'dv'$, where $|u'|_d = 0$, then the $\mathcal{M}_1$-computation starts with the component automaton M_3, which executes the cycle $w = cu'dv' \vdash^{\mathrm{c}}_{M_3} u'dv'$, after which automaton M_4 becomes

active. Now M_4 can erase the leftmost occurrence of the letter d only if $|u'|_b = 0$, which, however, is satisfied if $w \in L$. It will then execute the cycle $u'dv' \vdash^c_{M_4} u'v'$.

Now by repeatedly cycling through these two cycles of length two, the word w will be reduced to the empty word, if it is an element of the language L, and then automaton M_+ is called, which accepts.

Conversely, we see from the definition of $\mathcal{M}_1$ that all accepting computations proceed as described above, which implies that only words from the language L_1 are accepted. It follows that $L_{=1}(\mathcal{M}_1) = L_1$ holds. □

Now let L_2 be the language $L_{abc} = \{ w \in \{a,b,c\}^* \mid |w|_a = |w|_b = |w|_c \}$ from Proposition 4. Then L_2 is accepted by the stl-det-local-CD-R(1)-system

$$\mathcal{M}_2 = ((M_a, \sigma_a), (M_b, \sigma_b), (M_c, \sigma_c), (M_+, \sigma_+)), \{a, +\})$$

given in the proof of Proposition 4. If we construct a stl-det-local-CD-R(1)-system $\mathcal{M}$ by combining the systems $\mathcal{M}_1$ and $\mathcal{M}_2$, replacing each occurrence of M_+ in the successor sets of $\mathcal{M}_1$ by M_a, then the resulting system will certainly accept all words from the product $L_1 \cdot L_2$. However, it will also execute the following accepting computation:

$$\begin{aligned} acdcbba \vdash^c_{M_1} cdcbba \vdash^c_{M_2} cdcba \vdash^c_{M_3} dcba \vdash^c_{M_4} cba \\ \vdash^c_{M_a} cb \vdash^c_{M_b} c \vdash^c_{M_c} \varepsilon \vdash^*_{M_+} \mathsf{Accept}. \end{aligned}$$

However, the word $acdcbba$ does not belong to the product $L_1 \cdot L_2$, a contradiction.

The problem in the above example results from the fact that, in computations of the system $\mathcal{M}_1$, the component automaton M_2 reads across occurrences of the symbol d when looking for the leftmost occurrence of the symbol b. Accordingly, it may delete an occurrence of b that does not belong to the first factor. Thus, we need to modify the system $\mathcal{M}_1$ into an equivalent system $\mathcal{M}'_1$ that completely deletes the word $u \in L_1$ in an accepting computation without deleting any letter from v, given a word uv as input, where $u \in L_1$ and $v \in \Sigma^+$. That is, $\mathcal{M}'_1$ must guess the last letter, say x, of u and erase u completely, making sure that none of its delete operations is executed to the right of the rightmost occurrence of x in u.

So let $\mathcal{M}_1 = ((M_i, \sigma_i)_{i \in I \cup \{+\}}, I_0)$ be a stl-det-local-CD-R(1)-system in normal form that accepts a language $L_1 \subseteq \Sigma^*$. We assume that M_+ is the only accepting component automaton of $\mathcal{M}_1$, and that this component only accepts the empty word, that is, δ_+ is defined as $\delta_+ = \{(¢, \mathsf{MVR}), (\$, \mathsf{Accept})\}$. Further, for each $i \in I$, we use $\Sigma_1^{(i)}$ and $\Sigma_2^{(i)}$ to denote the subalphabets of Σ that correspond to automaton M_i according to Definition 1. Finally we assume that the alphabet Σ is ordered. For simplicity we write $\Sigma = \{a_1, \ldots, a_n\}$, and call a_i the i-th letter of Σ, $1 \leq i \leq n$.

The stl-det-local-CD-R(1)-system $\mathcal{M}_1$ will now be modified into an equivalent system $\mathcal{M}'_1$ that meets the requirements stated above. This system will consist

of a (large) number of subsystems each of which is a (slightly) revised version of $\mathcal{M}_1$. These subsystems will be indexed by the set of n-tuples

$$\mathrm{IND} = \{\, (i_1, \ldots, i_n) \mid i_1, \ldots, i_n \in \{2, 1, 0, d\} \,\}.$$

Below we will describe the necessary modifications for the various subsystems, but as a first general rule we require that all those component automata of $\mathcal{M}_1$ are excluded from the subsystem $\mathcal{M}_1^{(i_1,\ldots,i_n)}$ that attempt to erase a letter a_s for which $i_s = 0$ or $i_s = d$ holds. With a word $w \in \Sigma^*$ we associate an index $\mathrm{IND}(w) = (i_1, \ldots, i_n)$ by taking

$$i_j = \left\{ \begin{array}{l} 2\,, \text{ if } |w|_{a_j} \geq 2 \\ 1\,, \text{ if } |w|_{a_j} = 1 \\ 0\,, \text{ if } |w|_{a_j} = 0 \end{array} \right\}$$

for all $j = 1, \ldots, n$.

Given a word $w \in \Sigma^*$ as input, $\mathcal{M}_1'$ guesses a tuple $\mathrm{IND}'(w) = (i_1, \ldots, i_n) \in \{2, 1, 0\}^n$, and then one of the initial component automata $M_k^{\mathrm{IND}'(w)}$ of subsystem $\mathcal{M}_1^{\mathrm{IND}'(w)}$ is activated. It attempts to erase the leftmost occurrence of a letter a_s for some s such that $i_s \neq 0$. If $\mathrm{IND}'(w) \neq \mathrm{IND}(w)$, then at some point in the resulting computation this will be realized, causing $\mathcal{M}_1'$ to halt and reject (see the detailed description of $\mathcal{M}_1'$ below).

If $i_s = 2$, then $M_k^{\mathrm{IND}'(w)}$ transforms the word $w = w_1 a_s w_2$ into the word $w_1 w_2$, provided that $w_1 \in {\Sigma_1^{(k)}}^*$. Now $\mathrm{IND}(w_1 w_2)$ either coincides with $\mathrm{IND}(w)$ or it is obtained from $\mathrm{IND}(w)$ by replacing i_s by the value $i_s' = 1$. Accordingly, if $j \in \sigma_k$, then $j^{(i_1,\ldots,i_n)}, j^{(i_1,\ldots,i_s',\ldots,i_n)} \in \sigma_k^{(i_1,\ldots,i_n)}$.

If $i_s = 1$, then M_k would transform the word $w = w_1 a_s w_2$ into the word $w_1 w_2$, provided that $w_1 \in {\Sigma_1^{(k)}}^*$. Here $|w_1 w_2|_{a_s} = 0$, if $\mathrm{IND}'(w) = \mathrm{IND}(w)$, that is, within this cycle M_k would delete the last occurrence of the letter a_s. This, however, may cause problems (see the discussion above), and so we must be very carefull when simulating this step. If $w_1 w_2$ contains occurrences of letters that do not belong to the subalphabet $\Sigma_1^{(k)}$, then either the above cycle will not be completed successfully, if w_1 contains such a letter, or these letters are all contained in w_2, implying that a_s is not the rightmost letter of w. Thus, if the subalphabet

$$\mathrm{Alph}(\mathrm{IND}'(w)) = \{\, a_j \in \Sigma \mid j \in \{1, \ldots, n\} \text{ such that } i_j \in \{2, 1\} \,\}$$

is not contained in $\Sigma_1^{(k)} \cup \{a_s\}$, then $M_k^{\mathrm{IND}'(w)}$ just simulates the above cycle of M_k, and $j^{(i_1,\ldots,0,\ldots,i_n)} \in \sigma_k^{(i_1,\ldots,i_n)}$ for all $j \in \sigma_k$.

If, however, $\mathrm{Alph}(\mathrm{IND}'(w)) \subseteq \Sigma_1^{(k)} \cup \{a_s\}$, then instead of $M_k^{\mathrm{IND}'(w)}$ a component automaton $M_j^{(i_1,\ldots,d,\ldots,i_n)}$ is activated for some $j \in \sigma_k$. Here the indicator d in position s means that the current word w contains a single occurrence of the letter a_s, but that in the corresponding computation of the system $\mathcal{M}_1$, this

letter has already been erased. Observe that the above property only depends on the value $\mathrm{IND}'(w)$ guessed and on the component automaton $M_k^{\mathrm{IND}'(w)}$, and hence, the set of initial components of $\mathcal{M}_1'$ can be chosen accordingly.

The component $M_j^{(i_1,\ldots,d,\ldots,i_n)}$ is obtained from M_j by applying the following modifications, where we assume that $\Sigma_2^{(j)} = \{a_r\}$. We see from the general rule above that $i_r \in \{2,1\}$ holds. Let $D(i_1,\ldots,i_n) = \{\, j \in \{1,\ldots,n\} \mid i_j = d \,\}$. Then $M_j^{(i_1,\ldots,d,\ldots,i_n)}$ consists of $|D(i_1,\ldots,i_n)|+1$ many subcomponents. For each $l \in D(i_1,\ldots,i_n)$, there is a component $M_{j,l}^{(i_1,\ldots,i_n)}$ that deletes an occurrence of the letter a_l if it is the first letter of the tape inscription, that is, the corresponding transition function is defined by $\delta_{\cdot}(\mathfrak{c}) = \mathsf{MVR}$ and $\delta_{\cdot}(a_l) = \varepsilon$. Further, the only successor system of $M_{j,l}^{(i_1,\ldots,d,\ldots,i_n)}$ is the system $M_j^{(i_1,\ldots,0,\ldots,i_n)}$, where $i_l = d$ is replaced by $i_l' = 0$, indicating that the last occurrence of the letter a_l has now been erased. Finally there is a subcomponent $M_{j,0}^{(i_1,\ldots,i_n)}$ that simulates the actual behaviour of M_j. Here we have to distinguish two cases.

If $i_r = 2$, then $M_{j,0}^{(i_1,\ldots,d,\ldots,i_n)}$ simply deletes the first occurrence of the letter a_r, and in doing so it may move across all letters from

$$(\Sigma_1^{(j)} \cap \mathrm{Alph}(i_1,\ldots,i_n)) \cup \{\, a_\mu \mid \mu \in D(i_1,\ldots,i_n) \,\},$$

that is, it may move across all letters in $a_l \in \Sigma_1^{(j)}$ for which the indicator i_l is 1 or 2, and across all letters a_l, for which $i_l = d$. Further, $p^{(i_1,\ldots,2,\ldots,d\ldots,i_n)}, p^{(i_1,\ldots,1,\ldots,d,\ldots,i_n)} \in \sigma_{j,0}^{(i_1,\ldots,d,\ldots,i_n)}$ for all $p \in \sigma_j$, where the index 2 or 1 is in position r.

If $i_r = 1$, then the behaviour is similar, if there exists a letter in $\mathrm{Alph}(i_1,\ldots,i_n)$ that is not contained in $\Sigma_1^{(j)} \cup \{a_r\}$. In that case $p^{(i_1,\ldots,0,\ldots,d\ldots,i_n)} \in \sigma_{j,0}^{(i_1,\ldots,d,\ldots,i_n)}$ for all $p \in \sigma_j$, where the index 0 is in position r.

Finally, if $i_r = 1$ and $\mathrm{Alph}(i_1,\ldots,i_n) \subseteq \Sigma_1^{(j)} \cup \{a_r\}$, then instead of $M_{j,0}^{(i_1,\ldots,d,\ldots,i_n)}$ a component automaton $M_p^{(i_1,\ldots,d,\ldots,d,\ldots,i_n)}$ is activated for some $p \in \sigma_j$. Here the additional indicator d in position r means that the current word contains a single occurrence of the letter a_r, but that in the corresponding computation of the system $\mathcal{M}_1$, this letter has already been erased. Observe that the above property only depends on $(i_1,\ldots,d,\ldots,i_n)$ and on the component automaton M_j, and hence, the set of successor components of $M_k^{\mathrm{IND}'(w)}$ can be chosen accordingly.

Finally the accepting component M_+ is only called from a component $M_q^{(i_1,\ldots,i_n)}$ for which all but one of the indicators $i_1,\ldots,i_n$ are 0, the only non-zero indicator i_ν is 1 or d, and $M_q^{(i_1,\ldots,i_n)}$ deletes an occurrence of the letter a_ν.

These modifications are now applied to all component automata $M_j^{(i_1,\ldots,i_n)}$, where $(i_1,\ldots,i_n) \in \mathrm{IND}$ and $j \in I$.

This completes the description of the system $\mathcal{M}_1'$. Concerning the behaviour of this system we observe the following:

(1) For each word $w \in L_{=1}(\mathcal{M}_1)$, $\mathcal{M}'_1$ has an accepting mode = 1 computation, that is, $L_{=1}(\mathcal{M}_1) \subseteq L_{=1}(\mathcal{M}'_1)$.
(2) If during a computation of $\mathcal{M}'_1$ a component automaton $M_r^{(i_1,\ldots,i_n)}$ is activated such that, for some j, $i_j \in \{2, 1, d\}$, but no symbol a_j is on the tape, then i_j will never be set to 0, and accordingly, this computation fails.
(3) If during a computation of $\mathcal{M}'_1$ a component automaton $M_r^{(i_1,\ldots,i_n)}$ is activated such that, for some j, $i_j = 0$, but there are still occurrences of the symbol a_j on the tape, then these occurrences will not be erased, and accordingly, the computation fails. Thus, during a computation of $\mathcal{M}'_1$, if w is the current tape inscription, then in an accepting computation the correct value for $\mathrm{IND}(w)$ must be guessed.
(4) Each time the last occurrence of a letter a_j is erased, it is ensured that this occurrence is not the last letter of the given input word or that it is the first letter currently on the tape. Thus, the very last letter of the given input word can only be erased when it has become the very first letter on the tape, that is, when the rest of the word has already been erased completely.

From (2) and (3) it follows that $\mathcal{M}'_1$ can only accept words from the language $L_{=1}(\mathcal{M}_1)$, which together with (1) implies that $\mathcal{M}_1$ and $\mathcal{M}'_1$ are indeed equivalent. From (4) it follows that on input a word of the form uv, where $u \in L_1$ and $v \in \Sigma^+$, $\mathcal{M}'_1$ has a computation that erases the prefix u completely and then calls the final component automaton M_+ without scanning any prefix of v. Conversely, if $\mathcal{M}'_1$ has a computation that, starting with input uv, $u, v \in \Sigma^*$, erases the prefix u completely and then calls the final component automaton M_+, then $u \in L_1$, and during this computation $\mathcal{M}'_1$ does not scan any prefix of v.

Thus, if we now replace every occurrence of M_+ in the set of initial components and in the sets of successor components of $\mathcal{M}'_1$ by the initial components of a stl-det-local-CD-R(1)-system $\mathcal{M}_2$ accepting a language L_2, then we obtain a stl-det-local-CD-R(1)-system $\mathcal{M}$ for the language $L_1 \cdot L_2$. Hence, we have the following closure property.

Theorem 5. *The language class* $\mathcal{L}_{=1}$(stl-det-local-CD-R(1)) *is closed under product.*

As a consequence of the construction above also the following results are immediate.

Corollary 12. *The language class* $\mathcal{L}_{=1}$(stl-det-local-CD-R(1)) *is closed under Kleene-star and Kleene-plus.*

For showing that the class $\mathcal{L}_{=1}$(stl-det-local-CD-R(1)) is not closed under morphisms, we need a variant of the language

$$L_{ab} = \{\, w \in \{a, b\}^* \mid |w|_a = |w|_b \,\}.$$

In analogy to Proposition 4 it can be shown that L_{ab} is accepted by a CD-system of stateless deterministic R-automata working in mode = 1. Now consider the

morphism $\varphi : \{a,b\}^* \to \{a,b\}^*$ that is induced by $a \mapsto ab$ and $b \mapsto b$, and let L'_{ab} denote the language $\varphi(L_{ab})$. It is easily seen that $w \in L'_{ab}$ if and only if $|w|_b = 2 \cdot |w|_a$, and each occurrence of a letter a in w is immediately followed by an occurrence of the letter b.

Lemma 5. *The language L'_{ab} is not accepted by any* stl-det-local-CD-R(1)*-system working in mode $= 1$.*

Proof. Assume that $\mathcal{M} = ((M_i, \sigma_i)_{i \in I}, I_0)$ is a stl-det-local-CD-R(1)-system in normal form satisfying $L_{=1}(\mathcal{M}) = L'_{ab}$. If $i_0 \in I$ such that M_{i_0} executes accept instructions, then we see from Proposition 2 that $S(M_{i_0}) = \{\varepsilon\}$ must hold. Thus, any given non-empty word $w \in L'_{ab}$ must first be reduced to the empty word by executing $|w|$ many cycles, and then a component M_{i_0} is called which accepts.

Consider the words $w = b^n(ab)^n \in L'_{ab}$ and $w' = b^n ba(ab)^{n-1} \notin L'_{ab}$, where $n > 0$ is sufficiently large. The system $\mathcal{M}$ has an accepting mode $= 1$ computation for input w. In this computation, if an occurrence of the letter a is deleted before the prefix b^n has been deleted completely, that is, if this accepting computation can be written as

$$w = b^n(ab)^n \vdash^c_{M_{i_1}} \cdots \vdash^c_{M_{i_j}} b^{n-j}(ab)^n \vdash^c_{M_{i_{j+1}}} b^{n-j}b(ab)^{n-1} \vdash^*_{\mathcal{M}} \mathsf{Accept},$$

then $\mathcal{M}$ will also perform the following computation:

$$w' = b^n ba(ab)^{n-1} \vdash^c_{M_{i_1}} \cdots \vdash^c_{M_{i_j}} b^{n-j}ba(ab)^{n-1} \vdash^c_{M_{i_{j+1}}} b^{n-j}b(ab)^{n-1} \vdash^*_{\mathcal{M}} \mathsf{Accept}.$$

Thus, $\mathcal{M}$ will also accept the word w', a contradiction.

Hence, in an accepting mode $= 1$ computation of $\mathcal{M}$ on input w, no occurrence of the letter a is deleted before the prefix b^n has been deleted completely. If n is sufficiently large, then this accepting computation can be written as

$$\begin{aligned} w = b^n(ab)^n \quad &\vdash^{c^k}_{\mathcal{M}} \quad b^{n-k}(ab)^n \quad \vdash^c_{M_i} \quad b^{n-k-1}(ab)^n \\ &\vdash^{c^l}_{\mathcal{M}} \quad b^{n-k-l-1}(ab)^n \vdash^c_{M_i} b^{n-k-l-2}(ab)^n \\ &\vdash^{c^{n-k-l-2}}_{\mathcal{M}} \quad (ab)^n \quad \vdash^*_{\mathcal{M}} \quad \mathsf{Accept} \end{aligned}$$

for some index $i \in I$ and some numbers $k, l \leq |I|$. But then $\mathcal{M}$ would also perform the following accepting computation:

$$\begin{aligned} b^{n+l+1}(ab)^n \quad &\vdash^{c^k}_{\mathcal{M}} \quad b^{n-k+l+1}(ab)^n \vdash^c_{M_i} \quad b^{n-k+l}(ab)^n \\ &\vdash^{c^l}_{\mathcal{M}} \quad b^{n-k}(ab)^n \quad \vdash^c_{M_i} \quad b^{n-k-1}(ab)^n \\ &\vdash^{c^l}_{\mathcal{M}} \quad b^{n-k-l-1}(ab)^n \vdash^c_{M_i} b^{n-k-l-2}(ab)^n \\ &\vdash^{c^{n-k-l-2}}_{\mathcal{M}} \quad (ab)^n \quad \vdash^*_{\mathcal{M}} \quad \mathsf{Accept}. \end{aligned}$$

As $b^{n+l+1}(ab)^n \notin L'_{ab}$, this is again a contradiction. It follows that L'_{ab} is not accepted by any stl-det-local-CD-R(1)-system working in mode $= 1$. □

As $L_{ab} \in \mathcal{L}_{=1}$(stl-det-local-CD-R(1)), Lemma 5 has the following consequence.

Corollary 13. *The language class* $\mathcal{L}_{=1}(\mathsf{stl\text{-}det\text{-}local\text{-}CD\text{-}R(1)})$ *is not closed under* ε*-free morphisms.*

Concerning inverse morphisms we have the following preliminary result.

Proposition 12. *The language class* $\mathcal{L}_{=1}(\mathsf{stl\text{-}det\text{-}local\text{-}CD\text{-}R(1)})$ *is closed under inverse projections.*

Proof. Let $\mathcal{M} = ((M_i, \sigma_i)_{i \in I}, I_0)$ be a $\mathsf{stl\text{-}det\text{-}local\text{-}CD\text{-}R(1)}$-system on Σ accepting a language $L \subseteq \Sigma^*$ in mode $= 1$, let Γ be an alphabet that is disjoint from Σ, and let $\pi : (\Sigma \cup \Gamma)^* \to \Sigma^*$ be the projection that is induced by $a \mapsto a$ for all $a \in \Sigma$ and $b \mapsto \varepsilon$ for all $b \in \Gamma$. By L_π we denote the language

$$L_\pi = \pi^{-1}(L) = \{\, w \in (\Sigma \cup \Gamma)^* \mid \pi(w) \in L \,\}.$$

From $\mathcal{M}$ we construct a $\mathsf{stl\text{-}det\text{-}local\text{-}CD\text{-}R(1)}$-system $\mathcal{M}_\pi$ for L_π as follows:

$$\mathcal{M}_\pi = ((D_1, \sigma_{D_1}), (D_2, \sigma_{D_2}), (M_i, \sigma_i)_{i \in I}), I_0 \cup \{D_1\}).$$

The R-automata D_1 and D_2 are defined as follows:

$$\begin{aligned}
D_1 : (1)\ & \delta_{D_1}(¢) = \mathsf{MVR}, \\
(2)\ & \delta_{D_1}(a) = \mathsf{MVR} \quad \text{for all } a \in \Sigma, \\
(3)\ & \delta_{D_1}(b) = \varepsilon \quad\ \ \text{for all } b \in \Gamma, \\
D_2 : (4)\ & \delta_{D_2}(¢) = \mathsf{MVR}, \\
(5)\ & \delta_{D_2}(a) = \mathsf{MVR} \quad \text{for all } a \in \Sigma, \\
(6)\ & \delta_{D_2}(b) = \varepsilon \quad\ \ \text{for all } b \in \Gamma,
\end{aligned}$$

and $\sigma_{D_1} = \{D_2\} \cup I_0$ and $\sigma_{D_2} = \{D_1\} \cup I_0$.

Given an input $w \in (\Sigma \cup \Gamma)^*$, $\mathcal{M}_\pi$ first uses the component automata D_1 and D_2 to delete all occurrences of symbols from Γ, and then it checks whether the word obtained is accepted by $\mathcal{M}$. It follows that $L_{=1}(\mathcal{M}_\pi) = L_\pi$. □

However, the following general closure property is still open.

Open Problem 5. *In the language class* $\mathcal{L}_{=1}(\mathsf{stl\text{-}det\text{-}local\text{-}CD\text{-}R(1)})$ *closed under inverse morphisms?*

The application of an inverse projection π^{-1} to a language $L \subseteq \Sigma^*$ results in the shuffle of L with the free monoid Γ^*, where Γ is the set of letters mapped to ε by π. In fact, it can be shown that the language class $\mathcal{L}_{=1}(\mathsf{stl\text{-}det\text{-}local\text{-}CD\text{-}R(1)})$ is closed under *disjoint shuffle*, that is, if $L_1 \subseteq \Sigma^*$ and $L_2 \subseteq \Gamma^*$ are languages in $\mathcal{L}_{=1}(\mathsf{stl\text{-}det\text{-}local\text{-}CD\text{-}R(1)})$, where $\Sigma \cap \Gamma = \emptyset$, then the shuffle of L_1 and L_2 is also in this language class.

Open Problem 6. *Derive further closure and non-closure results for the language class* $\mathcal{L}_{=1}(\mathsf{stl\text{-}det\text{-}local\text{-}CD\text{-}R(1)})$. *In particular, is this class closed under reversal?*

Let Σ be a finite alphabet, and let $\overline{\Sigma} = \{ \overline{a} \mid a \in \Sigma \}$ be a copy of Σ such that $\Sigma \cap \overline{\Sigma} = \emptyset$. By $\overline{\ } : \Sigma^* \to \overline{\Sigma}^*$ we denote the morphism that replaces each letter $a \in \Sigma$ by its copy $\overline{a}$. Then the language $L_\Sigma := \{ \mathrm{sh}(w, \overline{w}) \mid w \in \Sigma^* \}$ is called the *twin shuffle language* over Σ. These twin shuffle languages are quite expressive as shown by the following classical result.

Proposition 13. [26] *For each recursively enumerable language $L \subseteq \Sigma_T^*$, there exist an alphabet Σ containing Σ_T and a regular language $R \subseteq (\Sigma \cup \overline{\Sigma})^*$ such that $L = \mathsf{Pr}^{\Sigma_T}(L_\Sigma \cap R)$.*

Observe that the twin shuffle language L_Σ is actually a rational trace language. Indeed, consider the dependency relation D_Σ on $\Sigma \cup \overline{\Sigma}$ that is defined by $D_\Sigma := \{ (a, b), (\overline{a}, \overline{b}) \mid a, b \in \Sigma \}$, and let $R_\Sigma := \{ a\overline{a} \mid a \in \Sigma \}^*$. Then R_Σ is a regular language over $\Sigma \cup \overline{\Sigma}$, and $[R_\Sigma]_{D_\Sigma} = L_\Sigma$. Hence, there exists a stl-det-local-CD-R(1)-system $\mathcal{M}_\Sigma$ satisfying $L_{=1}(\mathcal{M}_\Sigma) = L_\Sigma$. Accordingly, we obtain the following consequence.

Corollary 14. *For each recursively enumerable language $L \subseteq \Sigma_T^*$, there exist an alphabet Σ containing Σ_T, a language $L_1 \in \mathcal{L}_{=1}(\text{stl-det-local-CD-R}(1))$, and a regular language $R \subseteq (\Sigma \cup \overline{\Sigma})^*$ such that $L = \mathsf{Pr}^{\Sigma_T}(L_1 \cap R)$.*

Thus, we see that the closure of the language class $\mathcal{L}_{=1}(\text{stl-det-local-CD-R}(1))$ under intersection with regular sets and projections already yields all recursively enumerable languages.

7 Decision Problems

Each cycle of a deterministic restarting automaton can be simulated in linear time by a Turing machine. As each cycle is strictly length-reducing, it follows that a stl-det-local-CD-R(1)-system can be simulated by a nondeterministic Turing machine in quadratic time using linear space. In fact, a stl-det-local-CD-R(1)-system can be simulated by a nondeterministic shrinking RRWW-automaton, which yields the following result (see [10] and [18]).

Proposition 14. $\mathcal{L}_{=1}(\text{stl-det-local-CD-R}(1)) \subseteq \mathsf{NTIME}(n^2) \cap \mathsf{DSPACE}(n)$*, that is, the membership problem for the language $L_{=1}(\mathcal{M})$ of a* stl-det-local-CD-R(1)*-system $\mathcal{M}$ can be solved nondeterministically in quadratic time and deterministically in linear space.*

Theorem 2 yields an effective construction of a finite-state acceptor B from a stl-det-local-CD-R(1)-system $\mathcal{M}$ such that the language $E = L(B)$ is a subset of the language $L = L_{=1}(\mathcal{M})$ 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 15. *The following decision problems are effectively decidable:*

Instance *: A* stl-det-local-CD-R(1)*-system* $\mathcal{M}$.
Question 1 *: Is the language* $L_{=1}(\mathcal{M})$ *empty?*
Question 2 *: Is the language* $L_{=1}(\mathcal{M})$ *finite?*

Thus, the emptiness problem and the finiteness problem are effectively decidable for stl-det-local-CD-R(1)-systems. On the other hand, it is undecidable in general whether a rational trace language is recognizable (see, e.g., [7]). 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, it follows from Corollary 9 that it is undecidable in general whether a given stl-det-local-CD-R(1)-system accepts a regular language, that is, the following decision problem is undecidable in general.

Proposition 16. *The following decision problem is undecidable in general:*

Instance *: A* stl-det-local-CD-R(1)*-system* $\mathcal{M}$.
Question *: Is the language* $L_{=1}(\mathcal{M})$ *regular?*

Finally we consider the inclusion problem and the equivalence problem for stl-det-local-CD-R(1)-systems. We will see that these problems are also undecidable. For doing so we need the following notion.

A *rational transducer* is defined as $T = (Q, \Sigma, \Delta, q_0, F, E)$, where Q is a finite set of internal states, Σ is a finite input alphabet, Δ is a finite output alphabet, $q_0 \in Q$ is the initial state, $F \subseteq Q$ is the set of final states, and $E \subset Q \times \Sigma^* \times \Delta^* \times Q$ is a finite set of transitions.

If $e = (p_1, u_1, v_1, q_1)(p_2, u_2, v_2, q_2) \cdots (p_n, u_n, v_n, q_n) \in E^*$ is a sequence of transitions, then its *label* is the pair $\ell(e) = (u_1 u_2 \cdots u_n, v_1 v_2 \cdots v_n) \in \Sigma^* \times \Delta^*$. By $\ell_{\text{in}}(e)$ we denote the first component $u_1 u_2 \cdots u_n \in \Sigma^*$, and by $\ell_{\text{out}}(e)$ we denote the second component $v_1 v_2 \cdots v_n \in \Delta^*$. The sequence e above is called a *path* from p_1 to q_n, if $p_{i+1} = q_i$ for all $i = 1, \ldots, n-1$. It is called *successful* if p_1 is the initial state q_0, and if q_n is a final state. By $\Lambda(p, q)$ we denote the set of all paths from $p \in Q$ to $q \in Q$, and we define $\Lambda(p, Q') = \bigcup_{q \in Q'} \Lambda(p, q)$ for all subsets $Q' \subseteq Q$. Finally, $T(p, q) = \{ \ell(e) \mid e \in \Lambda(p, q) \}$ and $T(p, Q') = \{ \ell(e) \mid e \in \Lambda(p, Q') \}$. Thus, $\Lambda(q_0, F)$ is the set of all successful paths, and $T(q_0, F)$ is the set of labels of all successful paths. Then $\text{Rel}(T) = T(q_0, F)$ is called the *relation* defined by T. For $u \in \Sigma^*$ and $v \in \Delta^*$, $T(u) = \{ v \in \Delta^* \mid (u, v) \in T(q_0, F) \}$, and $T^{-1}(v) = \{ u \in \Sigma^* \mid (u, v) \in T(q_0, F) \}$. Obviously, the *domain* of $\text{Rel}(T)$ is the language $L(T) = \{ u \in \Sigma^* \mid T(u) \neq \emptyset \}$, which is the set of all input words for which T has an accepting computation.

As shown in Theorem 6.1 of [1] the relations defined by rational transducers are just the so-called *rational relations*, that is, the rational subsets of the monoid $\Sigma^* \times \Delta^*$. According to [8] Theorem 6.3 we have the following undecidability result.

Proposition 17. *The following version of the universality problem for rational transducers is undecidable in general:*

Instance : *A rational transducer* $T = (Q, \{a, b\}, \{c\}, q_0, F, E)$.
Question : *Is the relation* $\mathrm{Rel}(T)$ *universal, that is, does the equality* $\mathrm{Rel}(T) = \{a, b\}^* \times \{c\}^*$ *hold?*

The language $\hat{L} = \mathrm{sh}(\{a, b\}^*, \{c\}^*)$ is the rational trace language that is obtained from the regular language $R = \{a, b\}^* \cdot \{c\}^*$ and the dependency relation $D = \{(a, a), (b, b), (c, c), (a, b), (b, a)\}$ on the alphabet $\Sigma = \{a, b, c\}$. Hence, there exists a stl-det-local-CD-R(1)-system $\hat{\mathcal{M}}$ such that $L_{=1}(\hat{\mathcal{M}}) = \hat{L}$ by Theorem 3.

Now let $T = (Q, \{a, b\}, \{c\}, q_0, F, E)$ be a rational transducer. By introducing an intermediate state p_t for each transition of the form $t = (p, u, v, q)$ and by replacing t by the two transitions $t_i = (p, u, \varepsilon, p_t)$ and $t_o = (p_t, \varepsilon, v, q)$ we obtain a transducer that, in each step, either consumes part of its input or produces an output. Next we split each transition of the form $t_i = (p, u, \varepsilon, p_t)$, where $|u| > 1$, into $|u|$ many transitions, each of which just consumes a single letter, and we split each transition of the form $t_o = (p_t, \varepsilon, v, q)$, where $|v| > 1$, into $|v|$ many transitions that each produce just a single letter. The resulting transducer T' can now be viewed as a nondeterministic finite-state acceptor A' with ε-transitions on the alphabet $\Sigma = \{a, b, c\}$ by interpreting each transition of the form (p, x, ε, p') or (p, ε, x, p') as a transition (p, x, p'). It follows immediately from the above construction that the language $L' = L(A')$ accepted by A' has the following properties:

1. $L' \subseteq \bigcup_{(u,v) \in \mathrm{Rel}(T)} \mathrm{sh}(u, v)$, and
2. for all $(u, v) \in \mathrm{Rel}(T)$, there exists a word $w \in L'$ such that $\mathsf{Pr}^{\{a,b\}}(w) = u$ and $\mathsf{Pr}^{\{c\}}(w) = v$. Here $\mathsf{Pr}^{\{a,b\}} : \Sigma^* \to \{a, b\}^*$ denotes the projection onto $\{a, b\}^*$, and $\mathsf{Pr}^{\{c\}} : \Sigma^* \to \{c\}^*$ denotes the projection onto $\{c\}^*$.

From A' we can construct a deterministic finite-state acceptor A for the language L', and from A we obtain a stl-det-local-CD-R(1)-system $\mathcal{M}$ such that $L_{=1}(\mathcal{M}) = \bigcup_{w \in L'} [w]_D$ by the construction given in the proof of Theorem 3. Now we have the following chain of equivalences:

$$\begin{aligned} L_{=1}(\mathcal{M}) = L_{=1}(\hat{\mathcal{M}}) \ &\text{iff} \ \textstyle\bigcup_{w \in L'} [w]_D = \mathrm{sh}(\{a, b\}^*, \{c\}^*) \\ &\text{iff} \ \mathrm{Rel}(T) = \{a, b\}^* \times \{c\}^*. \end{aligned}$$

As the system $\mathcal{M}$ is effectively constructed from the given transducer T, Proposition 17 yields the following undecidability results.

Proposition 18. *The following decision problems are undecidable in general:*

Instance : *Two* stl-det-local-CD-R(1)*-systems* $\mathcal{M}_1$ *and* $\mathcal{M}_2$.
Question 1 : *Is* $L_{=1}(\mathcal{M}_1)$ *contained in* $L_{=1}(\mathcal{M}_2)$*?*
Question 2 : *Are* $\mathcal{M}_1$ *and* $\mathcal{M}_2$ *equivalent, that is, does* $L_{=1}(\mathcal{M}_1) = L_{=1}(\mathcal{M}_2)$ *hold?*

8 Concluding Remarks

We have seen that the stateless deterministic R-automata induce an infinite hierarchy of language classes based on the size of their windows, and we have related this hierarchy to the classical language families of regular and (deterministic) context-free languages. In [12] stateless variants of deterministic RR-automata have been introduced and studied. It remains to investigate the influence of the size of the read/write window on the expressive power of these automata. This also holds for the nondeterministic variants of stateless R- and RR-automata.

We have then seen that the stl-det-local-CD-R(1)-systems accept a subclass of all semi-linear languages that contains all rational trace languages, but that this subclass is incomparable to the (deterministic) linear languages and context-free languages. However, it remains open whether this language class can be characterized through other, more traditional, means. Also closure or non-closure of the language class $\mathcal{L}_{=1}$(stl-det-local-CD-R(1)) under certain operations like inverse morphisms or reversal are still open.

Further, it remains to determine the trade-off between stl-det-local-CD-R(1)-systems on the one hand and (deterministic or nondeterministic) finite-state acceptors on the other hand. Also it remains to study the exact degree of complexity for those decision problems that we have shown to be solvable for stl-det-local-CD-R(1)-systems. Finally, one could also study CD-systems of nondeterministic stateless CD-R(1)-systems. Are they more expressive than their locally deterministic counterparts considered in this paper?

References

1. J. Berstel. *Transductions and Context-Free Languages*, Leitfäden der angewandten Mathematik und Mechanik, vol. 38 (Teubner Studienbücher: Informatik), Teubner, Stuttgart, 1979.
2. J.-C. Birget. Intersection and union of regular languages and state complexity. *Inform. Proc. Letters* 43 (1992) 185–190.
3. G. Buntrock. *Wachsende kontext-sensitive Sprachen*. Habilitationsschrift, Fakultät für Mathematik und Informatik, Universität Würzburg, 1996.
4. E. Csuhaj-Varjú, J. Dassow, J. Kelemen, and G. Păun. *Grammar Systems. A Grammatical Approach to Distribution and Cooperation*, Gordon and Breach, London, 1994.
5. E. Csuhaj-Varjú, C. Martín-Vide, and V. Mitrana. Multiset automata. In: C.S. Calude, G. Păun, G. Rozenberg, and A. Salomaa (eds.), *Multiset Processing*, *Lect. Notes Comput. Sci. 2235*, Springer, Berlin, 2001, 69-83.
6. J. Dassow, G. Păun, and G. Rozenberg. Grammar systems. In: G. Rozenberg and A. Salomaa (eds.), *Handbook of Formal Languages*, Vol. 2, Springer, Berlin, 1997, 155–213.
7. V. Diekert and G. Rozenberg (eds.), *The Book of Traces*, World Scientific, Singapore, 1995.
8. O. Ibarra. Reversal-bounded multicounter machines and their decision problems. *J. Assoc. Comput. Mach.* 25 (1978) 116–133.

9. P. Jančar, F. Mráz, M. Plátek, and J. Vogel. Restarting automata. In: H. Reichel (ed.), *FCT 1995, Proc.*, *Lect. Notes Comput. Sci. 965*, Springer, Berlin, 1995, 283–292.
10. T. Jurdziński and F. Otto. Shrinking restarting automata. *Int. J. Found. Comput. Sci.* 18 (2007) 361–385.
11. M. Kutrib, H. Messerschmidt, and F. Otto. On stateless two-pushdown automata and restarting automata. In: E. Csuhaj-Varjú and Z. Ésik (eds.), *Automata and Formal Languages, AFL 2008, Proc.*, Computer and Automation Research Institute, Hungarian Academy of Sciences, 2008, 257–268.
12. M. Kutrib, H. Messerschmidt, and F. Otto. On stateless deterministic restarting automata. In: M. Nielsen, A. Kučera, P.B. Miltersen, C. Palamidessi, P. Tuma, and F. Valencia (eds.), *SOFSEM 2009: Theory and Practice of Computer Science, Proc.*, *Lect. Notes Comput. Sci. 5404*, Springer, Berlin, 2009, 353–364.
13. M. Kutrib, H. Messerschmidt and F. Otto. On stateless two-pushdown automata and restarting automata. Extended version of [11]. *Int. J. Found. Comput. Sci.*, to appear.
14. C. Lautemann. One pushdown and a small tape. In: K. Wagner (ed.), *Dirk Siefkes zum 50. Geburtstag*, Technische Universität Berlin and Universität Augsburg, 1988, 42–47.
15. M. Lopatková, M. Plátek, and P. Sgall. Towards a formal model for functional generative description, analysis by reduction and restarting automata. *The Prague Bulletin of Mathematical Linguistics* 87 (2007) 1–20.
16. R. McNaughton, P. Narendran, and F. Otto. Church-Rosser Thue systems and formal languages, *Journal of the ACM* 35 (1988) 324–344.
17. H. Messerschmidt and F. Otto. On nonforgetting restarting automata that are deterministic and/or monotone. In: D. Grigoriev, J. Harrison, and E.A. Hirsch (eds.), *CSR 2006, Proc.*, *Lect. Notes Comput. Sci. 3967*, Springer, Berlin, 2006, 247–258.
18. H. Messerschmidt and F. Otto. Cooperating distributed systems of restarting automata. *Int. J. Found. Comput. Sci.* 18 (2007) 1333–1342.
19. H. Messerschmidt and F. Otto. Strictly deterministic CD-systems of restarting automata. In: E. Csuhaj-Varjú and Z. Ésik (eds.), *FCT 2007, Proc.*, *Lect. Notes Comput. Sci. 4639*, Springer, Berlin, 2007, 424–434.
20. H. Messerschmidt and F. Otto. On deterministic CD-systems of restarting automata. *Int. J. Found. Comput. Sci.* 20 (2009) 185–209.
21. H. Messerschmidt and H. Stamer. Restart-Automaten mit mehreren Restart-Zuständen. In: H. Bordihn (ed.), *Workshop "Formale Methoden in der Linguistik" und 14. Theorietag "Automaten und Formale Sprachen", Proc.*, Institut für Informatik, Universität Potsdam, 2004, 111–116.
22. K. Oliva, P. Květoň, and R. Ondruška. The computational complexity of rule-based part-of-speech tagging. In: V. Matousek and P. Mautner (eds.), *TSD 2003, Proc.*, *Lect. Notes Comput. Sci. 2807*, Springer, Berlin, 2003, 82–89.
23. F. Otto. Restarting automata and their relations to the Chomsky hierarchy. In: Z. Esik and Z. Fülöp (eds.), *DLT 2003, Proc.*, *Lect. Notes Comput. Sci. 2710*, Springer, Berlin, 2003, 55-74.
24. F. Otto. Restarting automata. In: Z. Ésik, C. Martin-Vide, and V. Mitrana (eds.), *Recent Advances in Formal Languages and Applications*, Studies in Computational Intelligence Vol. 25, Springer, Berlin, 2006, 269–303.
25. M. Plátek, M. Lopatková, and K. Oliva. Restarting automata: Motivations and applications. In: M. Holzer (ed.), *Workshop "Petrinets" und 13. Theorietag "Au-*

tomaten und Formale Sprachen", Institut für Informatik, Technische Universität München, Garching, 2003, 90–96.
26. A. Salomaa. *Jewels of Formal Language Theory.* Computer Science Press, Rockville, Maryland, 1981.