International Journal of Computer Mathematics
Vol. 00, No. 00, January 2008, 1–25

RESEARCH ARTICLE

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

Benedek Nagy[a] and Friedrich Otto[b+]

[a]*Department of Computer Science, Faculty of Informatics, University of Debrecen, 4032 Debrecen, Egyetem tér 1., Hungary*
[b]*Fachbereich Elektrotechnik/Informatik, Universität Kassel, 34109 Kassel, Germany*

(*v3.6 released September 2008*)

It is known that cooperating distributed systems (CD-systems) of stateless deterministic restarting automata with window size 1 accept a class of semi-linear languages that properly includes all rational trace languages. Although the component automata of such a CD-system are all deterministic, the CD-system itself is not. Here we study CD-systems of stateless deterministic restarting automata with window size 1 that are themselves completely deterministic. In fact, we consider two such types of CD-systems, the *strictly deterministic* systems and the *globally deterministic* systems.

Keywords: restarting automaton; cooperating distributed system; determinism; language class; rational trace language

AMS Subject Classification: 68Q45

1. Introduction

Cooperating distributed systems (CD-systems) of restarting automata have been defined in [8] as an adaptation of the notion of *CD-grammar system with external control* [1, 2] to the setting of restarting automata. As expected CD-systems of restarting automata are much more expressive than their component automata. In [9, 10] also various types of *deterministic* CD-systems of restarting automata have been introduced and studied. As it turned out, even these CD-systems are quite expressive, but fairly complicated.

On the other hand, a simplified variant of restarting automata, the so-called *stateless restarting automata*, have been considered in [5, 6]. These are restarting automata with only a single state. 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 restarting automata are in general much less expressive than their

*This work was supported by grants from the Balassi Intézet Magyar Ösztöndíj Bizottsága (MÖB) and the Deutsche Akademischer Austauschdienst (DAAD). The first author was also supported by the TÁMOP 4.2.1/B-09/1/KONV-2010-0007 project, which is implemented through the New Hungary Development Plan, co-financed by the European Social Fund and the European Regional Development Fund. The results of this paper have been presented at the 5-th International Conference on Language and Automata Theory and Applications (LATA 2011) in Tarragona, Spain, May 2011. An extended abstract appeared in the proceedings of that conference [14].
+Corresponding author. Email: otto@theory.informatik.uni-kassel.de

ISSN: 0020-7160 print/ISSN 1029-0265 online

DOI: 0020716YYxxxxxxxx
http://www.informaworld.com

corresponding counterparts with states. As stateless restarting automata without auxiliary symbols are a very simple type of computing device that is easily implemented, it is a natural and interesting question to investigate the increase in computational power that is obtained by combining several automata of this type into a cooperating distributed system. Accordingly CD-systems of these weaker devices have been introduced and studied, a line of research that has been initiated in [11] by studying CD-systems of stateless deterministic restarting automata that have a read/write window of size 1 (so-called stl-det-R(1)-automata).

While the stl-det-R(1)-automata themselves have a severely restricted expressive power, it turned out that by combining several such automata into a CD-system a device is obtained that is surprizingly expressive. Actually in [11] only CD-systems of stl-det-R(1)-automata are considered that work in *mode* = 1, that is, the active component automaton executes a single cycle only, and then a successor component is chosen to continue with the computation (see the detailed definition below). In fact, it is shown in [11] that in mode = 1 these systems accept all rational trace languages. Further, the class of languages that are accepted by mode = 1 computations of CD-systems of stl-det-R(1)-automata is closed under union, product, Kleene star, commutative closure, and disjoint shuffle, but it is not closed under intersection with regular languages, complementation, or ε-free morphisms. In addition, the emptiness and the finiteness problems are easily solvable for these CD-systems, while their regularity, inclusion, and equivalence problems are in general undecidable [12].

A major feature of these CD-systems is the fact that, although all their component automata are deterministic, the CD-system itself is not, as in each of its computations, the initial component and the successor components are still chosen nondeterministically. Actually, as pointed out in [13] these CD-systems correspond to *nondeterministic finite-state acceptors with translucent letters*. Accordingly it is only natural to search for a type of CD-system that corresponds to *deterministic finite-state acceptors with translucent letters*, and to investigate how these CD-systems are related to the ones considered in [11]. Here we present such a type of CD-system of stl-det-R(1)-automata in that we present CD-systems of stl-det-R(1)-automata that are themselves completely deterministic. Actually, following the development in [9, 10] we introduce two different kinds of deterministic CD-systems: the *strictly deterministic* systems and the *globally deterministic* systems.

In a *strictly deterministic* system, there is only a single initial component, and each component automaton has only a single successor component. This ensures that all computations of such a CD-system are completely deterministic, but at the same time it severely restricts the expressive power of these systems. As we will see these systems do not even accept all finite languages.

We then concentrate on *globally deterministic* systems, which also have a single initial component only, but for which the successor component of a stl-det-R(1)-automaton is chosen based on the symbol that is being deleted in the current cycle. This still guarantees that each computation of a globally deterministic CD-system is completely deterministic, but it allows for much more flexibility. In fact, it is this type of deterministic CD-system of stl-det-R(1)-automata that corresponds to the deterministic finite-state acceptors *with translucent letters* of [13]. We study the class of languages that are accepted by these globally deterministic CD-systems of stl-det-R(1)-automata in quite some detail. We compare this class of languages to the class of rational trace languages and other well-known language families, we study its closure and nonclosure properties, and we investigate some of its algorithmic properties in short. For example, we will see that globally deterministic CD-systems of stl-det-R(1)-automata accept all regular languages, but they are not

as expressive as the CD-systems of stl-det-R(1)-automata considered in [11], as they do not accept all rational trace languages.

This paper is structured as follows. In Section 2 we give the definition of the stl-det-R(1)-automaton and of the stl-det-local-CD-R(1)-system from [11], and we restate some of the main results on these systems. In Section 3 we define the strictly deterministic CD-systems of stl-det-R(1)-automata, and we show that they have a rather weak expressive power. In addition, we prove that the class of languages accepted by these systems is an *anti-AFL* that is not even closed under reversal; however, this language class is closed under complementation. Then in Section 4, we define the main notion of this paper, the globally deterministic CD-system of stl-det-R(1)-automata (stl-det-global-CD-R(1)-system, for short). We show that these systems accept all regular languages, we present a normal form result for them, and we prove that they are not sufficiently expressive to accept all rational trace languages. Thus, they are strictly less expressive than the locally deterministic systems of [11]. Also we show that the class of languages accepted by the globally deterministic CD-systems of stl-det-R(1)-automata is closed under complementation, but that it is not closed under union, intersection with regular languages, product, Kleene star, reversal, or commutation. Thus, with respect to closure properties these systems are much weaker than the locally deterministic systems. Finally we turn to decision problems for stl-det-global-CD-R(1)-systems in Section 5. While the decidability of the membership, emptiness, and finiteness problems follows immediately from the corresponding results for stl-det-local-CD-R(1)-systems, the closure under complementation implies that also the *universe problem* is decidable for stl-det-global-CD-R(1)-systems. This is an important contrast to the situation for stl-det-local-CD-R(1)-systems, where the regularity, inclusion, and equivalence problems are shown to be undecidable by a reduction from the universe problem. Here we present a reduction from the Post Correspondence Problem to show that the inclusion problem is still undecidable for stl-det-global-CD-R(1)-systems. The paper closes with a short summary and some open problems in Section 6.

2. CD-Systems of Stateless Deterministic R(1)-Automata

Stateless types of restarting automata were introduced in [5] (see also [7]). Here we are only interested in the most restricted form of them, the *stateless deterministic* R-*automaton* of window size 1 (that is, the stl-det-R(1)-automaton). A stl-det-R(1)-*automaton* is a one-tape machine that is described by a 5-tuple $M = (\Sigma, ¢, \$, 1, \delta)$, where Σ is a finite (input) alphabet, the symbols $¢, \$ \notin \Sigma$ serve as markers for the left and right border of the work space, respectively, the size of the *read/write window* is 1, and $\delta : \Sigma \cup \{¢, \$\} \to \{\mathsf{MVR}, \mathsf{Accept}, \varepsilon\}$ is a (partial) *transition function*. There are three types of transition steps: *move-right steps* (MVR), which shift the window one step to the right, combined *rewrite/restart steps* (denoted by ε), which delete the content a of the window, thereby shortening the tape, and place the window over the left end of the tape, and *accept steps* (Accept), which cause the automaton to halt and accept. Finally we use the notation $\delta(a) = \emptyset$ to express the fact that the function δ is undefined for the symbol a. Some additional restrictions apply in that the sentinels ¢ and \$ must not be deleted, and that the window must not move right on seeing the \$-symbol.

A *configuration* of M is described by a pair (α, β), where either $\alpha = \varepsilon$ (the empty word) and $\beta \in \{¢\} \cdot \Sigma^* \cdot \{\$\}$ or $\alpha \in \{¢\} \cdot \Sigma^*$ and $\beta \in \Sigma^* \cdot \{\$\}$; here $\alpha\beta$ is the current content of the tape, and it is understood that the window contains the first symbol of β. A *restarting configuration* is of the form $(\varepsilon, ¢w\$)$, where $w \in \Sigma^*$; to simplify

the notation a restarting configuration $(\varepsilon, \mathtt{¢}w\$)$ is usually simply written as $\mathtt{¢}w\$$. By $\vdash_M$ we denote the single-step computation relation of M, and $\vdash_M^*$ denotes the reflexive transitive closure of $\vdash_M$.

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

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

$$\begin{array}{ll} (1.)\ \Sigma_{\mathrm{M}} = \{\, a \in \Sigma \mid \delta(a) = \mathsf{MVR} \,\}, & (3.)\ \Sigma_{\mathrm{A}} = \{\, a \in \Sigma \mid \delta(a) = \mathsf{Accept} \,\}, \\ (2.)\ \Sigma_{\varepsilon} = \{\, a \in \Sigma \mid \delta(a) = \varepsilon \,\}, & (4.)\ \Sigma_{\emptyset} = \{\, a \in \Sigma \mid \delta(a) = \emptyset \,\}. \end{array}$$

It has been observed in [11] that the language $L(M)$ can be characterized as follows:

$$L(M) = \begin{cases} \emptyset, & \text{if } \delta(\mathtt{¢}) = \emptyset, \\ \Sigma^*, & \text{if } \delta(\mathtt{¢}) = \mathsf{Accept}, \\ (\Sigma_{\mathrm{M}} \cup \Sigma_{\varepsilon})^* \cdot \Sigma_{\mathrm{A}} \cdot \Sigma^*, & \text{if } \delta(\mathtt{¢}) = \mathsf{MVR} \text{ and } \delta(\$) \neq \mathsf{Accept}, \\ (\Sigma_{\mathrm{M}} \cup \Sigma_{\varepsilon})^* \cdot ((\Sigma_{\mathrm{A}} \cdot \Sigma^*) \cup \{\varepsilon\}), & \text{if } \delta(\mathtt{¢}) = \mathsf{MVR} \text{ and } \delta(\$) = \mathsf{Accept}. \end{cases}$$

Cooperating distributed systems (CD-systems) of restarting automata were introduced and studied in [8]. Here we study restricted variants of the CD-systems of stl-det-$\mathsf{R}(1)$-automata of [11].

A (locally deterministic) CD-system of stl-det-$\mathsf{R}(1)$-automata, denoted as a stl-det-local-CD-$\mathsf{R}(1)$-*system*, consists of a finite collection $\mathcal{M} = ((M_i, \sigma_i)_{i \in I}, I_0)$. Here I is a finite *index set*, for each $i \in I$, the component $M_i = (\Sigma, \mathtt{¢}, \$, 1, \delta_i)$ is a stl-det-$\mathsf{R}(1)$-automaton and $\sigma_i \subseteq I$ is the *set of successors* for component M_i, and $I_0 \subseteq I$ is a set of *initial indices*. Here it is required that $I_0 \neq \emptyset$, and that $\sigma_i \neq \emptyset$ for all $i \in I$. In [11] it was required in addition that $i \notin \sigma_i$ for all $i \in I$, but this requirement is easily met by using two isomorphic copies of each component automaton. Therefore, we abondon it here in order to simplify the presentation.

As for CD-grammar systems (see, e.g., [1, 2]) various modes of operation have been introduced and studied for CD-systems of restarting automata, but here we are only interested in mode $= 1$ computations. A computation of $\mathcal{M}$ in mode $= 1$ on an input word w proceeds as follows. First an index $i_0 \in I_0$ is chosen nondeterministically. Then the stl-det-$\mathsf{R}(1)$-automaton M_{i_0} starts the computation with the initial configuration $\mathtt{¢}w\$$, and executes a single cycle. Thereafter an index $i_1 \in \sigma_{i_0}$ is chosen nondeterministically, and M_{i_1} continues the computation by executing a single cycle. This continues until, for some $l \geq 0$, the automaton M_{i_l} accepts. Such a computation will be denoted as

$$(i_0, w) \vdash_{\mathcal{M}}^c (i_1, w_1) \vdash_{\mathcal{M}}^c \cdots \vdash_{\mathcal{M}}^c (i_l, w_l) \vdash_{M_{i_l}}^* \mathsf{Accept}.$$

Should at some stage the chosen automaton 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 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. By $\mathcal{L}_{=1}(\mathsf{stl\text{-}det\text{-}local\text{-}CD\text{-}R(1)})$ we denote the class of languages that are accepted by mode $= 1$ computations of stl-det-local-CD-R(1)-systems. These CD-systems are called *locally deterministic*, as each component automaton is deterministic, but obviously, the system $\mathcal{M}$ as such is still nondeterministic.

Example 2.1 Let $\mathcal{M} = ((M_i, \sigma_i)_{i\in I}, I_0)$, where $\Sigma = \{a, b, c\}$, $I = \{a, b, c\}$, $I_0 = \{a\}$, $\sigma_a = \{b\}$, $\sigma_b = \{c\}$, $\sigma_c = \{a\}$, and M_a, M_b, and M_c are the stateless deterministic R(1)-automata that are given by the following transition functions:

$$\begin{array}{llllll} M_a : & \delta_a(¢) = \mathsf{MVR}, & \delta_a(a) = \varepsilon, & \delta_a(b) = \emptyset, & \delta_a(c) = \emptyset, & \delta_a(\$) = \mathsf{Accept}, \\ M_b : & \delta_b(¢) = \mathsf{MVR}, & \delta_b(a) = \mathsf{MVR}, & \delta_b(b) = \varepsilon, & \delta_b(c) = \mathsf{MVR}, & \delta_b(\$) = \emptyset, \\ M_c : & \delta_c(¢) = \mathsf{MVR}, & \delta_c(a) = \mathsf{MVR}, & \delta_c(b) = \mathsf{MVR}, & \delta_c(c) = \varepsilon, & \delta_c(\$) = \emptyset. \end{array}$$

The automaton M_a accepts the empty word. If the input is non-empty, then M_a deletes the first letter, provided it is an a; otherwise, it gets stuck, and so it rejects. The automaton M_b simply deletes the first occurrence of the letter b, and M_c simply deletes the first occurrence of the letter c. Thus, for each occurrence of a, also an occurrence of b and an occurrence of c is deleted. However, while M_b and M_c can read across occurrences of the letter a, M_a can read across neither b nor c. Hence, $L_{=1}(\mathcal{M})$ is the language $L_{abc} = \{\, w \in \{a, b, c\}^* \mid |w|_a = |w|_b = |w|_c \geq 0, \text{ and for each prefix } u \text{ of } w : |u|_a \geq \max\{|u|_b, |u|_c\} \,\}$. Obviously, this language is not context-free, as $L_{abc} \cap (a^* \cdot b^* \cdot c^*) = \{\, a^n b^n c^n \mid n \geq 0 \,\}$.

If $\Sigma = \{a_1, \ldots, a_n\}$, then the corresponding *Parikh mapping* $\psi : \Sigma^* \to \mathbb{N}^n$ is defined by $\psi(w) = (|w|_{a_1}, \ldots, |w|_{a_n})$. Recall from [3] or from [11] that a language $L \subseteq \Sigma^*$ is called (the linearization of) a *rational trace language* if there exists a reflexive and transitive binary relation D on Σ (a *dependency relation*) such that $L = \bigcup_{w\in R} [w]_D$ for some regular language R on Σ. Here $[w]_D$ denotes the congruence class of w with respect to the congruence $\equiv_D = \{\, (uabv, ubav) \mid u, v \in \Sigma^*, a, b \in \Sigma, (a, b) \notin D \,\}$.

Proposition 2.2 [11]

(a) *Each language* $L \in \mathcal{L}_{=1}(\mathsf{stl\text{-}det\text{-}local\text{-}CD\text{-}R(1)})$ *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.
(b) $\mathcal{L}_{=1}(\mathsf{stl\text{-}det\text{-}local\text{-}CD\text{-}R(1)})$ *properly contains the class of all rational trace languages, and therewith it contains all regular languages.*

It follows from Proposition 2.2 (a) that $\mathcal{L}_{=1}(\mathsf{stl\text{-}det\text{-}local\text{-}CD\text{-}R(1)})$ only contains languages that are semi-linear, that is, languages with semi-linear Parikh image. 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 L, we see from (a) that this language is not accepted by any stl-det-local-CD-R(1)-system. Together with Example 2.1 this implies that the language class $\mathcal{L}_{=1}(\mathsf{stl\text{-}det\text{-}local\text{-}CD\text{-}R(1)})$ is incomparable to the classes DLIN, LIN, DCFL, and CFL with respect to inclusion, where DLIN denotes the class of *deterministic linear languages*, which is the class of languages that are accepted by deterministic one-turn pushdown automata, LIN is the class of *linear languages*, and DCFL and CFL denote the classes of *deterministic context-free* and *context-free languages*.

3. Strictly Deterministic CD-R(1)-Systems

Although all the component automata of a stl-det-local-CD-R(1)-system are deterministic, the system itself is not. Indeed the initial component with which to begin a particular computation is chosen nondeterministically from the set I_0 of all initial components, and after each cycle the component for executing the next cycle is chosen nondeterministically from among all the successors of the previously active component. Observe that in deriving the main results of [11] this feature is used repeatedly in essential ways. Here we introduce and study a type of CD-system of stl-det-R(1)-automata that is completely deterministic. The idea and the notation is taken from [9], where a corresponding notion was introduced for CD-systems of general restarting automata.

A CD-system $\mathcal{M} = ((M_i, \sigma_i)_{i \in I}, I_0)$ of stl-det-R(1)-automata is called *strictly deterministic* if $|I_0| = 1$ and $|\sigma_i| = 1$ for all $i \in I$. Then, for each word $w \in \Sigma^*$, $\mathcal{M}$ has a unique computation that begins with the initial configuration corresponding to input w. Thus, $\mathcal{M}$ is completely deterministic. By $\mathcal{L}_{=1}$(stl-det-strict-CD-R(1)) we denote the class of languages that are accepted by strictly deterministic CD-systems of stl-det-R(1)-automata working in mode = 1.

Observe that the CD-system in Example 2.1 is strictly deterministic. On the other hand, we have the following negative result.

LEMMA 3.1 *The finite language $L_0 = \{aaa, bb\}$ is not accepted by any* stl-det-strict-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-strict-CD-R(1)-system such that $L_{=1}(\mathcal{M}) = L_0$, let $I_0 = \{i_0\}$, and let $\sigma_{i_0} = \{i_1\}$. Obviously, $\delta_{i_0}(¢) = \mathsf{MVR}$, and $\delta_{i_0}(a) = \delta_{i_0}(b) = \varepsilon$. Now $(i_0, aaa) \vdash^c_{\mathcal{M}} (i_1, aa)$, which leads to acceptance, while $(i_0, baa) \vdash^c_{\mathcal{M}} (i_1, aa)$ should lead to rejection, which is a contradiction. Thus, L_0 is not accepted by any stl-det-strict-CD-R(1)-system working in mode = 1. □

Thus, we obtain the following immediate consequences.

COROLLARY 3.2 *The language class $\mathcal{L}_{=1}$(stl-det-strict-CD-R(1)) is incomparable under inclusion to the language classes* FIN *of finite languages,* REG *of regular languages, and* CFL *of context-free languages. In particular, it follows that the inclusion $\mathcal{L}_{=1}$(stl-det-strict-CD-R(1)) $\subseteq$ $\mathcal{L}_{=1}$(stl-det-local-CD-R(1)) is proper.*

From Lemma 3.1 we immediately obtain several nonclosure properties for the class $\mathcal{L}_{=1}$(stl-det-strict-CD-R(1)). In fact, we have the following result.

THEOREM 3.3 *The language class $\mathcal{L}_{=1}$(stl-det-strict-CD-R(1)) is an anti-AFL, that is, it is not closed under union, product, Kleene plus, intersection with regular sets, ε-free morphisms, and inverse morphisms.*

Proof. It is easily seen that the languages $\{aaa\}$, $\{bb\}$, and $\{a, b\}^*$ are accepted by stl-det-strict-CD-R(1)-systems. As $\{aaa\} \cup \{bb\} = \{aaa, bb\} = \{aaa, bb\} \cap \{a, b\}^*$, Lemma 3.1 shows that this language class is neither closed under union nor under intersection with regular sets.

The languages $\{c, d\}$ and $\{c^6\}$ are accepted by stl-det-strict-CD-R(1)-systems. Let $h_1 : \{c, d\}^* \to \{a, b\}^*$ be the morphism defined by $c \mapsto aaa$ and $d \mapsto bb$, and let $h_2 : \{a, b\}^* \to \{c\}^*$ be the morphism defined by $a \mapsto c^2$ and $b \mapsto c^3$. Then $h_1(\{c, d\}) = \{aaa, bb\} = h_2^{-1}(\{c^6\})$, and hence, Lemma 3.1 shows that this language class is neither closed under ε-free morphisms nor under inverse morphisms.

For showing nonclosure under product we consider the languages $\{a\}^*$ and $\{b\}^*$,

which are accepted by stl-det-strict-CD-R(1)-systems.

Claim 1. $L_{\text{prod}} = \{a\}^* \cdot \{b\}^* \notin \mathcal{L}_{=1}(\text{stl-det-strict-CD-R}(1))$.

Proof. Assume that $\mathcal{M} = ((M_i, \sigma_i)_{i\in I}, I_0)$ is a stl-det-strict-CD-R(1)-system such that $L_{=1}(\mathcal{M}) = L_{\text{prod}}$, let $I_0 = \{i_0\}$, and let $\sigma_{i_0} = \{i_1\}$. Obviously, $\delta_{i_0}(\text{¢}) = \text{MVR}$, and as $\mathcal{M}$ must accept all powers of a, $\delta_{i_0}(a)$ cannot be undefined. Analogously, as $\mathcal{M}$ must accept all powers of b, $\delta_{i_0}(b)$ cannot be undefined, either. Further, $\delta_{i_0}(a) \neq \text{Accept} \neq \delta_{i_0}(b)$.

If $\delta_{i_0}(a) = \text{MVR} = \delta_{i_0}(b)$, then $\delta_{i_0}(\$) = \text{Accept}$ would follow, which would imply that $L_{=1}(\mathcal{M}) = \{a, b\}^*$ holds, a contradiction.

If $\delta_{i_0}(a) = \text{MVR}$ and $\delta_{i_0}(b) = \varepsilon$, then the computation of $\mathcal{M}$ on input ab would start with the cycle $(i_0, ab) \vdash^c_{\mathcal{M}} (i_1, a)$, and the computation of $\mathcal{M}$ on input ba would start with the cycle $(i_0, ba) \vdash^c_{\mathcal{M}} (i_1, a)$. As $ab \in L_{\text{prod}}$, while $ba \notin L_{\text{prod}}$, this contradicts our assumption on $L_{=1}(\mathcal{M})$.

If $\delta_{i_0}(a) = \varepsilon$ and $\delta_{i_0}(b) = \text{MVR}$, then the computation of $\mathcal{M}$ on input ab would start with the cycle $(i_0, ab) \vdash^c_{\mathcal{M}} (i_1, b)$, and the computation of $\mathcal{M}$ on input ba would start with the cycle $(i_0, ba) \vdash^c_{\mathcal{M}} (i_1, b)$, which yields the same contradiction.

Finally, if $\delta_{i_0}(a) = \varepsilon = \delta_{i_0}(b)$, then $\mathcal{M}$ could not distinguish between the words aa and ba. As this covers all cases, we see that L_{prod} is not accepted by any stl-det-strict-CD-R(1)-system. □

For showing nonclosure under Kleene plus we consider the language $L_s = \{\, ab^n \mid n \geq 1 \,\}$, which is easily seen to be accepted by a stl-det-strict-CD-R(1)-system.

Claim 2. $L_{\text{plus}} = (L_s)^* \notin \mathcal{L}_{=1}(\text{stl-det-strict-CD-R}(1))$.

Proof. Assume that $\mathcal{M} = ((M_i, \sigma_i)_{i\in I}, I_0)$ is a stl-det-strict-CD-R(1)-system such that $L_{=1}(\mathcal{M}) = L_{\text{plus}}$, let $I_0 = \{i_0\}$, and let $\sigma_{i_0} = \{i_1\}$, $\sigma_{i_1} = \{i_2\}$, and $\sigma_{i_2} = \{i_3\}$.

First we consider the component automaton M_{i_0}. Obviously, $\delta_{i_0}(\text{¢}) = \text{MVR}$, and $\delta_{i_0}(a)$ is defined. If $\delta_{i_0}(a) = \text{Accept}$, then $L_{=1}(\mathcal{M}) = a \cdot \{a, b\}^*$. So assume that $\delta_{i_0}(a) = \text{MVR}$. If $\delta_{i_0}(b)$ is undefined, then $L_{=1}(\mathcal{M}) = \{a\}^*$ or $L_{=1}(\mathcal{M}) = \emptyset$, if $\delta_{i_0}(b) = \text{Accept}$, then $L_{=1}(\mathcal{M}) = \{a\}^* \cdot b \cdot \{a, b\}^*$, and if $\delta_{i_0}(b) = \text{MVR}$, then $L_{=1}(\mathcal{M}) = \{a, b\}^*$ or $L_{=1}(\mathcal{M}) = \emptyset$. Finally, if $\delta_{i_0}(b) = \varepsilon$, then $\mathcal{M}$ executes the cycles $(i_0, ab) \vdash^c_{\mathcal{M}} (i_1, a)$ and $(i_0, ba) \vdash^c_{\mathcal{M}} (i_1, a)$. However, $ab \in L_{\text{plus}}$, while $ba \notin L_{\text{plus}}$. Hence, it follows that $\delta_{i_0}(a) = \varepsilon$.

Next we consider M_{i_1}. Obviously, $\delta_{i_1}(\text{¢}) = \text{MVR}$, and $\delta_{i_1}(b)$ is defined. If $\delta_{i_1}(b) = \text{Accept}$, then $L_{=1}(\mathcal{M}) \supseteq ab \cdot \{a, b\}^*$. So assume that $\delta_{i_1}(b) = \text{MVR}$. Then $\delta_{i_1}(a)$ must be defined. If $\delta_{i_1}(a) = \text{Accept}$, then $\mathcal{M}$ accepts all words that have a prefix of the form $ab^m a$. If $\delta_{i_1}(a) = \text{MVR}$, then either $\mathcal{M}$ accepts all words with first letter a, or it does not accept any of these words. Finally, if $\delta_{i_1}(a) = \varepsilon$, then $\mathcal{M}$ executes the cycles $(i_1, bab) \vdash^c_{\mathcal{M}} (i_2, bb)$ and $(i_1, abb) \vdash^c_{\mathcal{M}} (i_2, bb)$. However, $abab \in L_{\text{plus}}$, while $aabb \notin L_{\text{plus}}$. Hence, it follows that $\delta_{i_1}(b) = \varepsilon$.

Finally, we consider M_{i_2}. Obviously, $\delta_{i_2}(\text{¢}) = \text{MVR}$, and $\delta_{i_2}(a)$ and $\delta_{i_2}(b)$ are defined. It is easily seen that $\delta_{i_2}(a) \neq \text{Accept} \neq \delta_{i_2}(b)$ hold. Assume that $\delta_{i_2}(b) = \text{MVR}$. If also $\delta_{i_2}(a) = \text{MVR}$, then either $\mathcal{M}$ accepts all words with prefix ab, or it does not accept any of these words. On the other hand, if $\delta_{i_2}(a) = \varepsilon$, then $\mathcal{M}$ executes the cycles $(i_2, ab) \vdash^c_{\mathcal{M}} (i_3, b)$ and $(i_2, ba) \vdash^c_{\mathcal{M}} (i_3, b)$. However, $abab \in L_{\text{plus}}$, while $abba \notin L_{\text{plus}}$. Hence, it follows that $\delta_{i_2}(b) = \varepsilon$. If $\delta_{i_2}(a) = \text{MVR}$, then $\mathcal{M}$ executes the cycles $(i_2, ab) \vdash^c_{\mathcal{M}} (i_3, a)$ and $(i_2, ba) \vdash^c_{\mathcal{M}} (i_3, a)$. However, $abab \in L_{\text{plus}}$, while $abba \notin L_{\text{plus}}$. Finally, if $\delta_{i_2}(a) = \varepsilon$, then $\mathcal{M}$ executes the cycles $(i_2, bab) \vdash^c_{\mathcal{M}} (i_3, ab)$ and $(i_2, aab) \vdash^c_{\mathcal{M}} (i_3, ab)$. Since $abbab \in L_{\text{plus}}$, while

$abaab \notin L_{\text{plus}}$, this yields a contradiction as well.

As this covers all cases, we see that L_{plus} is not accepted by any stl-det-strict-CD-R(1)-system. □

This completes the proof that the language class $\mathcal{L}_{=1}$(stl-det-strict-CD-R(1)) is an anti-AFL. □

If $((M_i, \sigma_i)_{i\in I}, I_0)$ is a stl-det-strict-CD-R(1)-system for a language $L \subseteq \Sigma^*$, then by turning undefined transition steps into Accept steps and vice versa, we obtain a stl-det-strict-CD-R(1)-system for the language $L^c = \Sigma^* \setminus L$. This yields our only closure property for stl-det-strict-CD-R(1)-systems.

PROPOSITION 3.4 *The language class* $\mathcal{L}_{=1}$(stl-det-strict-CD-R(1)) *is closed under the operation of complementation.*

We close this section with two additional nonclosure properties.

PROPOSITION 3.5

(a) *The language class* $\mathcal{L}_{=1}$(stl-det-strict-CD-R(1)) *is not closed under the operation of reversal.*
(b) *The language class* $\mathcal{L}_{=1}$(stl-det-strict-CD-R(1)) *is not closed under the operation of taking the commutative closure.*

Proof. (a) Let $\Sigma = \{a, b\}$, and let $L_a = \{\, ab^n \mid n \geq 0 \,\}$. Then L_a is easily seen to be accepted by the stl-det-strict-CD-R(1)-system. Now we consider the language $L_a^R = \{\, b^n a \mid n \geq 0 \,\}$.

Claim 1. $L_a^R \notin \mathcal{L}_{=1}$(stl-det-strict-CD-R(1)).

Proof. Assume that $\mathcal{M}' = ((M_i', \sigma_i')_{i\in I}, \{i_0\})$ is a stl-det-strict-CD-R(1)-system on $\Sigma = \{a, b\}$ such that $L_{=1}(\mathcal{M}') = L_a^R$. First we analyze the initial component M_{i_0}' of $\mathcal{M}'$.

If $\delta_{i_0}'(¢) = \emptyset$, then $L_{=1}(\mathcal{M}') = \emptyset$ would follow, and if $\delta_{i_0}'(¢) = \mathsf{Accept}$, then $L_{=1}(\mathcal{M}') = \Sigma^*$ would follow. Thus, we see that $\delta_{i_0}'(¢) = \mathsf{MVR}$ holds.

As $a \in L_a^R$ and $ba \in L_a^R$, $\delta_{i_0}'(a)$ and $\delta_{i_0}'(b)$ must both be defined. On the other hand, $aa \notin L_a^R$ and $b \notin L_a^R$, which means that $\delta_{i_0}'(a) \neq \mathsf{Accept} \neq \delta_{i_0}'(b)$.

Now assume that $\delta_{i_0}'(a) = \mathsf{MVR}$. As $a \in L_a^R$, this implies that $\delta_{i_0}'(\$) = \mathsf{Accept}$. But then $\mathcal{M}'$ would also accept the word $aa \notin L_a^R$. Hence, it follows that $\delta_{i_0}'(a) = \varepsilon$. Let $\sigma_{i_0}' = \{i_1\}$, that is, M_{i_1}' is the unique successor component of M_{i_0}'. Then $(i_0, a) \vdash_{\mathcal{M}'}^c (i_1, \varepsilon) \vdash_{M_{i_1}}^* \mathsf{Accept}$, while $(i_0, aa) \vdash_{\mathcal{M}'}^c (i_1, a)$, and the configuration (i_1, a) must not lead to acceptance. Hence, we see that $\delta_{i_1}'(¢) = \mathsf{MVR}$, $\delta_{i_1}'(\$) = \mathsf{Accept}$, and $\delta_{i_1}'(a) \neq \mathsf{MVR}$.

Next assume that $\delta_{i_0}'(b) = \mathsf{MVR}$. Then $\mathcal{M}'$ executes the cycle $(i_0, ba) \vdash_{\mathcal{M}'}^c (i_1, b)$, and as $ba \in L_a^R$, the configuration (i_1, b) leads to acceptance. However, $\mathcal{M}'$ also executes the cycle $(i_0, ab) \vdash_{\mathcal{M}'}^c (i_1, b)$, that is, it would also accept on input $ab \notin L_a^R$. Hence, it follows that $\delta_{i_0}'(b) = \varepsilon$. However, this yields the computation $(i_0, b) \vdash_{\mathcal{M}'}^c (i_1, \varepsilon) \vdash_{M_{i_1}}^* \mathsf{Accept}$, which also contradicts our assumption above as $b \notin L_a^R$. As this covers all possible cases, we conclude that L_a^R is not accepted by any stl-det-strict-CD-R(1)-system. □

Thus, the language L_a witnesses the fact that the language class $\mathcal{L}_{=1}$(stl-det-strict-CD-R(1)) is not closed under the operation of reversal.

(b) Let $\Sigma = \{a, b, c\}$, and let

$$L_c = \{\, a^n \mid n \geq 1 \,\} \cup \{\, awcz \mid w \in \{a,b\}^*,\ |w|_b \geq 1 + |w|_a,\ z \in \Sigma^* \,\}.$$

Claim 2. $L_c \in \mathcal{L}_{=1}(\mathsf{stl\text{-}det\text{-}strict\text{-}CD\text{-}R(1)})$.

Proof. Let $\mathcal{M} = ((M_i, \sigma_i)_{i\in\{0,1,2,3\}}, \{0\})$ be the stl-det-strict-CD-R(1)-system on Σ that is defined as follows:

$$\begin{array}{llll}
\delta_0(¢) = \mathsf{MVR}, & \delta_1(¢) = \mathsf{MVR}, & \delta_2(¢) = \mathsf{MVR}, & \delta_3(¢) = \mathsf{MVR},\\
\delta_0(a) = \varepsilon, & \delta_1(a) = \mathsf{MVR}, & \delta_2(a) = \varepsilon, & \delta_3(a) = \mathsf{MVR},\\
\delta_0(b) = \emptyset, & \delta_1(b) = \varepsilon, & \delta_2(b) = \mathsf{MVR}, & \delta_3(b) = \varepsilon,\\
\delta_0(c) = \emptyset, & \delta_1(c) = \emptyset, & \delta_2(c) = \mathsf{Accept}, & \delta_3(c) = \emptyset,\\
\delta_0(\$) = \emptyset, & \delta_1(\$) = \mathsf{Accept}, & \delta_2(\$) = \emptyset, & \delta_3(\$) = \emptyset,\\
\sigma_0 = \{1\}, & \sigma_1 = \{2\}, & \sigma_2 = \{3\}, & \sigma_3 = \{2\}.
\end{array}$$

Given a word $w \in \Sigma^*$ as input, the initial component M_0 checks that w is of the form $w = aw_1$. In the negative, it rejects; otherwise, the letter a is deleted and component M_1 becomes active. If $w_1 = a^n$ for some $n \geq 0$, then M_1 accepts; otherwise it looks for the first occurrence of b that must only be preceded by a's. If there is no such occurrence, then M_1 rejects; otherwise, this occurrence of the letter b is deleted and component M_2 becomes active. Now the components M_2 and M_3 delete occurrences of the letters a and b, respectively, until M_2 discovers an occurrence of c that is only preceded by b's, and then M_2 accepts. If no such c is encountered, or if there is no occurrence of b that is only preceded by a's when M_3 is active, then the computation fails. It now follows that $L_{=1}(\mathcal{M}) = L_c$. □

The commutative closure $\hat{L}_c$ of the language L_c is the language

$$\hat{L}_c = \{\, a^n \mid n \geq 1 \,\} \cup \{\, w \in \Sigma^* \mid |w|_a \geq 1,\ |w|_b \geq 1, \text{ and } |w|_c \geq 1 \,\}.$$

Claim 3. $\hat{L}_c \notin \mathcal{L}_{=1}(\mathsf{stl\text{-}det\text{-}strict\text{-}CD\text{-}R(1)})$.

Proof. Assume that $\mathcal{M}' = ((M'_i, \sigma'_i)_{i\in I}, \{i_0\})$ is a stl-det-strict-CD-R(1)-system on Σ satisfying $L_{=1}(\mathcal{M}') = \hat{L}_c$. Let us first analyze the starting component M'_{i_0} of $\mathcal{M}'$.

As $\emptyset \neq \hat{L}_c \neq \Sigma^*$, we see that $\delta'_{i_0}(¢) = \mathsf{MVR}$. Further, as $a^n \in \hat{L}_c$ for all $n \geq 1$, while $\varepsilon \notin \hat{L}_c$, we see that $\delta'_{i_0}(a) = \varepsilon$. Let $\sigma'_{i_0} = \{i_1\}$.

As $(i_0, a) \vdash^c_{\mathcal{M}'} (i_1, \varepsilon)$ and as $a \in \hat{L}_c$, while $ab \notin \hat{L}_c$, we conclude that $\delta'_{i_1}(¢) = \mathsf{MVR}$ and $\delta'_{i_1}(\$) = \mathsf{Accept}$.

Let us return to δ'_{i_0}. As $bac \in \hat{L}_c$ and $cba \in \hat{L}_c$, we see that $\delta'_{i_0}(b)$ and $\delta'_{i_0}(c)$ must be defined. On the other hand, as $b, c \notin \hat{L}_c$, we see that $\delta'_{i_0}(b), \delta'_{i_0}(c) \notin \{\mathsf{Accept}, \varepsilon\}$, either, that is, $\delta'_{i_0}(b) = \mathsf{MVR} = \delta'_{i_0}(c)$. Thus, on input a^{n+1}, $\mathcal{M}'$ executes the cycle $(i_0, a^{n+1}) \vdash^c_{\mathcal{M}'} (i_1, a^n)$, and on input $w = uav$, where $u \in \{b, c\}^*$, $\mathcal{M}'$ executes the cycle $(i_0, uav) \vdash^c_{\mathcal{M}'} (i_1, uv)$. Hence, we must now analyze the behaviour of M'_{i_1}.

As $aa, abc, acb \in \hat{L}_c$, we see that $\delta'_{i_1}(a)$, $\delta'_{i_1}(b)$, and $\delta'_{i_1}(c)$ are all defined. On the other hand, as $aab, ac, ab \notin \hat{L}_c$, we see that $\delta_{i_1}(x) \neq \mathsf{Accept}$ for all $x \in \Sigma$.

If $\delta'_{i_1}(b) = \mathsf{MVR}$, then $(i_1, b) \vdash^*_{M'_{i_1}} \mathsf{Accept}$, contradicting the fact that $ab \notin \hat{L}_c$. Analogously, if $\delta'_{i_1}(c) = \mathsf{MVR}$, then $(i_1, c) \vdash^*_{M'_{i_1}} \mathsf{Accept}$, contradicting the fact that $ac \notin \hat{L}_c$. Thus, we see that $\delta'_{i_1}(b) = \varepsilon = \delta'_{i_1}(c)$. Let $\sigma'_{i_1} = \{i_2\}$. Then $\mathcal{M}'$ will execute the sequence of cycles $(i_0, abc) \vdash^c_{\mathcal{M}'} (i_1, bc) \vdash^c_{\mathcal{M}'} (i_2, c)$, and the latter configuration

will lead to acceptance, as $abc \in \hat{L}_c$. But $\mathcal{M}'$ will also execute the sequence of cycles $(i_0, acc) \vdash^c_{\mathcal{M}'} (i_1, cc) \vdash^c_{\mathcal{M}'} (i_2, c)$, which means that (i_2, c) should not lead to acceptance, as $acc \notin \hat{L}_c$. It follows that the language $\hat{L}_c$ is not accepted by any stl-det-strict-CD-R(1)-system. □

Thus, we see that the language class $\mathcal{L}_{=1}$(stl-det-strict-CD-R(1)) is not closed under commutation. This completes the proof of Proposition 3.5. □

4. Globally Deterministic CD-R(1)-Systems

As the strictly deterministic CD-R(1)-systems do not even accept all finite languages, we now consider a less restricted variant of CD-systems of stateless deterministic R(1)-automata.

Let $((M_i, \sigma_i)_{i\in I}, I_0)$ be a CD-system of stl-det-R(1)-automata over Σ such that $|I_0| = 1$. For each $i \in I$, let $\Sigma^{(i)}_\varepsilon$ be the set of letters that are deleted by the component automaton M_i, and let $\Sigma^{(i)}_{\mathrm{M}}$ be the set of letters that the component automaton M_i can move across (see Section 2). Further, let $\delta : \bigcup_{i\in I}(\{i\} \times \Sigma^{(i)}_\varepsilon) \to I$ be a mapping that assigns to each pair $(i, a) \in \{i\} \times \Sigma^{(i)}_\varepsilon$ an element $j \in \sigma_i$. Then δ is called a *global successor function*. It assigns a successor component $j \in \sigma_i$ to the active component i based on the letter $a \in \Sigma^{(i)}_\varepsilon$ that is deleted by M_i in the current cycle. Thus, if $w = uav$, where $u \in \Sigma^{(i)*}_{\mathrm{M}}$ and $a \in \Sigma^{(i)}_\varepsilon$, and if $\delta(i, a) = j$, then $\mathcal{M}$ would execute the cycle $(i, ¢w\$) = (i, ¢uav\$) \vdash^c_{\mathcal{M}} (j, ¢uv\$)$. It follows that, for each input word $w \in \Sigma^*$, the system $\mathcal{M} = ((M_i, \sigma_i)_{i\in I}, I_0, \delta)$ has a unique computation that starts from the initial configuration corresponding to input w, that is, $\mathcal{M}$ is completely deterministic. Accordingly we call $\mathcal{M}$ a stl-det-global-CD-R(1)-system, and by $\mathcal{L}_{=1}$(stl-det-global-CD-R(1)) we denote the class of languages that are accepted by stl-det-global-CD-R(1)-systems working in mode = 1.

Obviously, each stl-det-strict-CD-R(1)-system is globally deterministic. However, the stl-det-global-CD-R(1)-systems are more expressive than the strictly deterministic ones.

Example 4.1 Let $\mathcal{M} = ((M_i, \sigma_i)_{i\in I}, I_0, \delta)$ be the stl-det-global-CD-R(1)-system over $\Sigma = \{a, b\}$ that is specified as follows:

$I = \{0, 1, 2, 3, +\}$, $I_0 = \{0\}$, $\sigma_0 = \{1, 3\}$, $\sigma_1 = \{2\}$, $\sigma_2 = \{+\} = \sigma_3$, $\sigma_+ = \{0\}$, and M_0, M_1, M_2, M_3, and M_+ are the stl-det-R(1)-automata that are given by the following transition functions:

$$\begin{aligned}
M_0 &: \delta_0(¢) = \mathsf{MVR},\ \delta_0(a) = \varepsilon,\ \delta_0(b) = \varepsilon,\ \delta_0(\$) = \emptyset,\\
M_1 &: \delta_1(¢) = \mathsf{MVR},\ \delta_1(a) = \varepsilon,\ \delta_1(b) = \emptyset,\ \delta_1(\$) = \emptyset,\\
M_2 &: \delta_2(¢) = \mathsf{MVR},\ \delta_2(a) = \varepsilon,\ \delta_2(b) = \emptyset,\ \delta_2(\$) = \emptyset,\\
M_3 &: \delta_3(¢) = \mathsf{MVR},\ \delta_3(a) = \emptyset,\ \delta_3(b) = \varepsilon,\ \delta_3(\$) = \emptyset,\\
M_+ &: \delta_+(¢) = \mathsf{MVR},\ \delta_+(a) = \emptyset,\ \delta_+(b) = \emptyset,\ \delta_+(\$) = \mathsf{Accept},
\end{aligned}$$

and δ is defined by $\delta(0, a) = 1$, $\delta(0, b) = 3$, $\delta(1, a) = 2$, $\delta(2, a) = +$, $\delta(3, b) = +$. Then it is easily seen that $L_{=1}(\mathcal{M}) = \{aaa, bb\}$, which is not accepted by any stl-det-strict-CD-R(1)-system working in mode = 1 by Lemma 3.1.

Thus, we have the following proper inclusion.

COROLLARY 4.2 $\mathcal{L}_{=1}$(stl-det-strict-CD-R(1)) $\subsetneq$ $\mathcal{L}_{=1}$(stl-det-global-CD-R(1)).

In fact, we also have the following proper inclusion.

LEMMA 4.3 $\mathsf{REG} \subsetneq \mathcal{L}_{=1}(\mathsf{stl\text{-}det\text{-}global\text{-}CD\text{-}R(1)})$.

Proof. From Example 2.1 we see that $\mathcal{L}_{=1}(\mathsf{stl\text{-}det\text{-}global\text{-}CD\text{-}R(1)})$ contains languages that are not even context-free. Thus, it remains to show that each regular language is accepted by a stl-det-global-CD-R(1)-system working in mode = 1.

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

- $I = Q$, $I_0 = \{p_0\}$, $\sigma_i = I$ for all $i \in I$,
- for each $i \in I$, the automaton M_i is defined through

$$\delta_i(\mathrm{¢}) = \mathsf{MVR},\ \delta_i(a) = \varepsilon \text{ for all } a \in \Sigma, \text{ and } \delta_i(\$) = \begin{cases} \mathsf{Accept}, & \text{if } i \in F, \\ \emptyset, & \text{otherwise,} \end{cases}$$

- and δ is defined through $\delta(i, a) = \delta_A(i, a)$ for all $i \in I$ and all $a \in \Sigma$.

By induction on $|w|$ it follows that, for all $w \in \Sigma^*$ and $i \in I$, $\delta_A(p_0, w) = i$ iff $(p_0, w) \vdash^{c^*}_{\mathcal{M}} (i, \varepsilon)$. Hence, $w \in L$ iff $\delta_A(p_0, w) \in F$ iff $(p_0, w) \vdash^{c^*}_{\mathcal{M}} (i, \varepsilon) \vdash^*_{M_i} \mathsf{Accept}$ iff $w \in L_{=1}(\mathcal{M})$, which shows that $L_{=1}(\mathcal{M}) = L$. Thus, each regular language is accepted by a stl-det-global-CD-R(1)-system working in mode = 1. □

To simplify the discussions and proofs below we now introduce a normal form for stl-det-global-CD-R(1)-systems.

DEFINITION 4.4 *Let $\mathcal{M} = ((M_i, \sigma_i)_{i\in I}, \{i_0\}, \delta)$ be a stl-det-global-CD-R(1)-system, and, for each $i \in I$, let $(\Sigma^{(i)}_{\mathrm{M}}, \Sigma^{(i)}_{\varepsilon}, \Sigma^{(i)}_{\mathrm{A}}, \Sigma^{(i)}_{\emptyset})$ be the partitioning of the underlying alphabet Σ that corresponds to the component automaton M_i (see Section 2). The system $\mathcal{M}$ is said to be in* normal form, *if it satisfies the following conditions:*

(1) Each component M_i is reachable from the initial component M_{i_0}, that is, for each $i \in I$, there exists an input $w \in \Sigma^*$ such that $(i_0, w) \vdash^{c^*}_{\mathcal{M}} (i, z)$ holds for some $z \in \Sigma^*$.
(2) For each component M_i, $\delta_i(\mathrm{¢}) = \mathsf{MVR}$.
(3) For each component M_i and each letter $a \in \Sigma$, $\delta_i(a) \in \{\mathsf{MVR}, \varepsilon\}$, that is, $\Sigma^{(i)}_{\mathrm{A}} = \emptyset = \Sigma^{(i)}_{\emptyset}$ for all $i \in I$.

Thus, if $\mathcal{M} = ((M_i, \sigma_i)_{i\in I}, \{i_0\}, \delta)$ is in *normal form*, then each computation of $\mathcal{M}$ ends with a component that accepts or rejects on the \$-symbol.

PROPOSITION 4.5
From a stl-det-global-CD-R(1)-system $\mathcal{M} = ((M_i, \sigma_i)_{i\in I}, \{i_0\}, \delta)$, a stl-det-global-CD-R(1)-system $\mathcal{M}' = ((M'_j, \sigma'_j)_{j\in J}, \{j_0\}, \delta')$ can be constructed such that $\mathcal{M}'$ is in normal form, and $L_{=1}(\mathcal{M}') = L_{=1}(\mathcal{M})$.

Proof. All those components of $\mathcal{M}$ that are not reachable from the initial component M_{i_0} can simply be deleted. By inspecting the successor function δ, these components can actually be determined. So we can now assume that all components of $\mathcal{M}$ are reachable from M_{i_0}.

Assume that $\delta_i(\mathrm{¢}) = \emptyset$ for some $i \in I$. Then each computation that reaches the component M_i gets stuck, and so it is rejecting. In particular, M_i never executes a rewrite step, and hence, the value of $\delta(i, a)$ ($a \in \Sigma$) is irrelevant. Define a new component M_- by $\delta_-(\mathrm{¢}) = \mathsf{MVR}$, $\delta_-(a) = \mathsf{MVR}$ for all $a \in \Sigma$, $\delta_-(\$) = \emptyset$, and

replace the component M_i by M_- in all successor sets and in the right-hand side of the function δ. Then the system obtained in this way still accepts the same language as $\mathcal{M}$.

If $\delta_i(\mathbb{c}) = \mathsf{Accept}$ for some $i \in I$, then each computation that reaches the component M_i accepts immediately. In particular, M_i never executes a rewrite step, and hence, the value of $\delta(i,a)$ $(a \in \Sigma)$ is irrelevant. Define a new component M_+ by $\delta_+(\mathbb{c}) = \mathsf{MVR}$, $\delta_+(a) = \mathsf{MVR}$ for all $a \in \Sigma$, $\delta_+(\$) = \mathsf{Accept}$, and replace the component M_i by M_+ in all successor sets and in the right-hand side of the function δ. Then the system obtained in this way still accepts the same language as $\mathcal{M}$. Thus, we may now assume that $\mathcal{M}$ satisfies conditions (1) and (2) from Definition 4.4.

Assume that now the system $\mathcal{M}$ has the form $\mathcal{M} = ((M_i,\sigma_i)_{i\in I}, \{i_0\}, \delta)$. We construct the system $\mathcal{M}' = ((M_i',\sigma_i)_{i\in I}, \{i_0\}, \delta')$ by revising, for each $i \in I$, the component M_i and the successor function δ as follows, where $a \in \Sigma$, and M_- and M_+ denote the components introduced above:

$$\begin{array}{lll}
\delta_i'(\mathbb{c}) = \mathsf{MVR}, & & \\
\delta_i'(a) = \mathsf{MVR}, & \text{if } \delta_i(a) = \mathsf{MVR}, & \\
\delta_i'(a) = \varepsilon, & \text{if } \delta_i(a) = \varepsilon, & \text{and } \delta'(i,a) = \delta(i,a),\\
\delta_i'(a) = \varepsilon, & \text{if } \delta_i(a) = \mathsf{Accept}, & \text{and } \delta'(i,a) = +,\\
\delta_i'(a) = \varepsilon, & \text{if } \delta_i(a) = \emptyset, & \text{and } \delta'(i,a) = -,\\
\delta_i'(\$) = \delta_i(\$). & &
\end{array}$$

Then $\mathcal{M}'$ is obviously in normal form.

Let $w \in \Sigma^*$. Then the computation of $\mathcal{M}$ on input w has the form $(i_0,w) \vdash_{\mathcal{M}}^{c^*} (i_r,w_r) \vdash_{M_{i_r}}^* (i_r,(\mathbb{c}u_r,v_r\$))$, and either $\delta_{i_r}(a) = \emptyset$, where a denotes the first letter of $v_r\$$, or $\delta_{i_r}(a) = \mathsf{Accept}$. In the former case $\mathcal{M}$ rejects on input w, while in the latter case it accepts. From the construction of $\mathcal{M}'$ we see that on input w, $\mathcal{M}'$ will execute the computation $(i_0,w) \vdash_{\mathcal{M}'}^{c^*} (i_r,w_r) \vdash_{M_{i_r}'}^* (i_r,(\mathbb{c}u_r,v_r\$))$. If $v_r\$ = \$$, then M_{i_r}' will reject or accept just as M_{i_r}, and if $v_r = ax_r$ for some letter $a \in \Sigma$, then M_{i_r}' will delete the letter a, and the component M_- or the component M_+ will become active. The former happens if $\delta_{i_r}(a) = \emptyset$, and the latter happens if $\delta_{i_r}(a) = \mathsf{Accept}$. Hence, we see that $\mathcal{M}'$ accepts on input w if and only if $\mathcal{M}$ does, that is, $L_{=1}(\mathcal{M}') = L_{=1}(\mathcal{M})$ holds. □

Using this normal form result the following inclusion can be derived easily.

PROPOSITION 4.6 $\mathcal{L}_{=1}(\mathsf{stl\text{-}det\text{-}global\text{-}CD\text{-}R(1)}) \subseteq \mathcal{L}_{=1}(\mathsf{stl\text{-}det\text{-}local\text{-}CD\text{-}R(1)})$.

Proof. Let $\mathcal{M} = ((M_i,\sigma_i)_{i\in I}, \{i_0\}, \delta)$ be a stl-det-global-CD-R(1)-system on Σ and, for each $i \in I$, let $(\Sigma_{\mathrm{M}}^{(i)}, \Sigma_{\varepsilon}^{(i)}, \Sigma_{\mathrm{A}}^{(i)}, \Sigma_{\emptyset}^{(i)})$ be the partitioning of the underlying alphabet Σ that corresponds to the component automaton M_i (see Section 2). By Proposition 4.5 we can assume that $\mathcal{M}$ is in normal form. From $\mathcal{M}$ we now construct a stl-det-local-CD-R(1)-system $\mathcal{M}' = ((M_j',\sigma_j')_{j\in J}, J_0)$ satisfying $L_{=1}(\mathcal{M}') = L_{=1}(\mathcal{M})$.

Let $J = \{\,(i,a) \mid i \in I,\ a \in \Sigma_{\varepsilon}^{(i)}\,\} \cup \{\,(i,+) \mid i \in I\,\}$, let $J_0 = \{\,(i_0,a) \mid a \in \Sigma_{\varepsilon}^{(i_0)}\,\} \cup \{(i_0,+)\}$, and take

$$\begin{array}{ll}
\sigma'_{(i,a)} = \{\,(j,b) \mid j = \delta(i,a),\ b \in \Sigma_{\varepsilon}^{(j)}\,\} \cup \{(\delta(i,a),+)\} & \text{for all } i \in I \text{ and } a \in \Sigma_{\varepsilon}^{(i)},\\
\sigma'_{(i,+)} = J & \text{for all } i \in I.
\end{array}$$

Finally, we define the stl-det-R(1)-automata $M'_{(i,a)}$ and $M'_{(i,+)}$ as follows, where

$i \in I$ and $a \in \Sigma_\varepsilon^{(i)}$:

$$\begin{array}{lll} M'_{(i,a)} : \delta'_{(i,a)}(\mathfrak{c}) = \mathsf{MVR}, & M'_{(i,+)} : \delta'_{(i,+)}(\mathfrak{c}) = \mathsf{MVR}, & \\ \quad \delta'_{(i,a)}(b) = \mathsf{MVR}, & \quad \delta'_{(i,+)}(b) = \mathsf{MVR} & \text{for all } b \in \Sigma_{\mathrm{M}}^{(i)}, \\ \quad \delta'_{(i,a)}(a) = \varepsilon, & \quad \delta'_{(i,+)}(a) = \emptyset, & \\ \quad \delta'_{(i,a)}(c) = \emptyset, & \quad \delta'_{(i,+)}(c) = \emptyset & \text{for all } c \in \Sigma_\varepsilon^{(i)} \setminus \{a\}, \\ \quad \delta'_{(i,a)}(\$) = \emptyset; & \quad \delta'_{(i,+)}(\$) = \delta_i(\$). & \end{array}$$

Let $w = a_1 a_2 \cdots a_n \in \Sigma^*$, where $n \geq 0$ and $a_1, \ldots, a_n \in \Sigma$. Assume that the computation of $\mathcal{M}$ on input w has the following form:

$$\begin{array}{l} (i_0, w) = (i_0, u_0 b_0 v_0) \vdash_{\mathcal{M}}^c (i_1, u_0 v_0) = (i_1, u_1 b_1 v_1) \vdash_{\mathcal{M}}^c \cdots \\ \qquad \vdash_{\mathcal{M}}^c (i_r, u_{r-1} v_{r-1}) = (i_r, w_r), \end{array}$$

and that starting with the configuration $(\varepsilon, \mathfrak{c} w_r \$)$, the automaton M_{i_r} performs a tail computation. Thus, $u_j \in \Sigma_{\mathrm{M}}^{(i_j)^*}$ and $b_j \in \Sigma_\varepsilon^{(i_r)}$ for all $j = 0, 1, \ldots, r-1$, and $w_r \in \Sigma_{\mathrm{M}}^{(i_r)^*}$. Then $\mathcal{M}'$ can execute the following sequence of cycles by guessing, in each step, what the next letter deleted by $\mathcal{M}$ will be:

$$\begin{array}{l} ((i_0, b_0), w) = ((i_0, b_0), u_0 b_0 v_0) \vdash_{\mathcal{M}'}^c ((i_1, b_1), u_0 v_0) = ((i_1, b_1), u_1 b_1 v_1) \vdash_{\mathcal{M}'}^c \\ \qquad \cdots \vdash_{\mathcal{M}'}^c ((i_r, +), u_{r-1} v_{r-1}) = ((i_r, +), w_r), \end{array}$$

and starting from the configuration $(\varepsilon, \mathfrak{c} w_r \$)$, $M'_{(i_r,+)}$ executes a tail computation that accepts if and only if the above tail computation of M_{i_r} accepts. Thus, we conclude that $L_{=1}(\mathcal{M}) \subseteq L_{=1}(\mathcal{M}')$ holds.

Conversely, if $\mathcal{M}'$ has an accepting computation on input $w \in \Sigma^*$, then it follows easily from the above construction of $\mathcal{M}'$ that $\mathcal{M}$ will also accept on input w. Thus, we see that $L_{=1}(\mathcal{M}') = L_{=1}(\mathcal{M})$, which completes the proof of Proposition 4.6. □

Is the inclusion $\mathcal{L}_{=1}(\mathsf{stl\text{-}det\text{-}global\text{-}CD\text{-}R(1)}) \subseteq \mathcal{L}_{=1}(\mathsf{stl\text{-}det\text{-}local\text{-}CD\text{-}R(1)})$ a strict one? Further, are all rational trace language already accepted by stl-det-global-CD-R(1)-systems? The following result anwers these questions.

PROPOSITION 4.7 *The rational trace language*

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

is not accepted by any stl-det-global-CD-R(1)*-system.*

Proof. The language $L_\vee$ is simply the commutative closure of the regular language $(ab)^* \cup (abb)^*$, and hence, it is a rational trace language with respect to the dependency relation $D = \{(a, a), (b, b)\}$ on $\Sigma = \{a, b\}$.

Claim. $L_\vee \notin \mathcal{L}_{=1}(\mathsf{stl\text{-}det\text{-}global\text{-}CD\text{-}R(1)})$.

Proof. Assume that $\mathcal{M} = ((M_i, \sigma_i)_{i \in I}, I_0, \delta)$ is a stl-det-global-CD-R(1)-system such that $L_{=1}(\mathcal{M}) = L_\vee$. Without loss of generality we can assume that $I = \{0, 1, \ldots, m-1\}$ and that $I_0 = \{0\}$.

Let $n > 2m$, and let $w = a^n b^n \in L_\vee$. Then the computation of $\mathcal{M}$ on input w is accepting, that is, it is of the form

$$(0, a^n b^n) \vdash_{\mathcal{M}}^c (i_1, w_1) \vdash_{\mathcal{M}}^c \cdots \vdash_{\mathcal{M}}^c (i_r, w_r) \vdash_{M_{i_r}}^* \mathsf{Accept},$$

where M_{i_r} accepts the tape contents $¢w_r\$$. If $|w_r|_a > 0$ and $|w_r|_b > 0$, then M_{i_r} would also accept the tape contents $w_r a^n b^{5m}$ for any $m \geq 0$, and therewith $\mathcal{M}$ would accept the input $wa^n b^{5n} = a^n b^n a^n b^{5n}$, which does not belong to $L_\vee$. Hence, it follows that $|w_r|_a = 0$ or $|w_r|_b = 0$. If $w_r = a^s$ for some $s > 0$, then it follows analogously that with w, $\mathcal{M}$ would also accept the word wa^m for all $m \geq 0$. Hence, it would accept the word $wa^n = a^n b^n a^n \notin L_\vee$.Thus, $|w_r|_a = 0$, and analogously it can be shown that $|w_r|_b = 0$, that is, $w_r = \varepsilon$. Hence, in the above computation $2n$ cycles are executed that delete the input $w = a^n b^n$ symbol by symbol, and then M_{i_r} accepts the empty word.

As $n > m$, there exists an index $i \in I$ and integers $s, t, k, \ell \geq 0$, $m \geq s + t \geq 0$ and $m \geq k + \ell > 0$, such that the above computation can be written as follows:

$$(0, a^n b^n) \vdash_{\mathcal{M}}^{c^*} (i, a^{n-s} b^{n-t}) \vdash_{\mathcal{M}}^{c^+} (i, a^{n-s-k} b^{n-t-\ell}) \vdash_{\mathcal{M}}^{c^*} (i_r, \varepsilon) \vdash_{M_{i_r}}^* \mathsf{Accept}.$$

Obviously, $\mathcal{M}$ can also execute the following shortened computation:

$$(0, a^{n-k} b^{n-\ell}) \vdash_{\mathcal{M}}^{c^*} (i, a^{n-s-k} b^{n-t-\ell}) \vdash_{\mathcal{M}}^{c^*} (i_r, \varepsilon) \vdash_{M_{i_r}}^* \mathsf{Accept},$$

that is, $\mathcal{M}$ accepts on input $a^{n-k} b^{n-\ell}$. From our assumption that $L_{=1}(\mathcal{M}) = L_\vee$ we can therefore conclude that $k = \ell$, as $n > 2m$.

Now consider the computation of $\mathcal{M}$ on input $a^n b^{2n}$. As $a^n b^{2n} \in L_\vee$, this computation is accepting, that is, it has the following form:

$$(0, a^n b^{2n}) \vdash_{\mathcal{M}}^{c^*} (i, a^{n-s} b^{2n-t}) \vdash_{\mathcal{M}}^{c^+} (i, a^{n-s-k} b^{2n-t-k}) \vdash_{\mathcal{M}}^{c^*} (i', \varepsilon) \vdash_{M_{i'}}^* \mathsf{Accept}.$$

But then $\mathcal{M}$ can also execute the following computation:

$$(0, a^{n-k} b^{2n-k}) \vdash_{\mathcal{M}}^{c^*} (i, a^{n-s-k} b^{2n-t-k}) \vdash_{\mathcal{M}}^{c^*} (i', \varepsilon) \vdash_{M_{i'}}^* \mathsf{Accept},$$

that is, it accepts on input $a^{n-k} b^{2n-k} \notin L_\vee$. Thus, $L_{=1}(\mathcal{M}) \neq L_\vee$, that is, $L_\vee$ is not accepted by any stl-det-global-CD-R(1)-system working in mode $= 1$. □

This completes the proof of Proposition 4.7. □

As all rational trace languages are accepted by stl-det-local-CD-R(1)-systems, we have the following consequence, which also answers the first of the above questions.

COROLLARY 4.8 $\mathcal{L}_{=1}(\text{stl-det-global-CD-R(1)}) \subsetneq \mathcal{L}_{=1}(\text{stl-det-local-CD-R(1)})$.

The Dyck language $D_1'^*$ is not a rational trace language, but it is accepted by the following stl-det-strict-CD-R(1)-system $\mathcal{M} = ((M_i, \sigma_i)_{i \in I}, I_0)$, where $I = \{a, b\}$, $I_0 = \{a\}$, $\sigma_a = \{b\}$, $\sigma_b = \{a\}$, and the stl-det-R(1)-automata M_a and M_b are defined by the following transition functions:

$$\begin{array}{l} M_a : \delta_a(¢) = \mathsf{MVR},\ \delta_a(a) = \varepsilon, \qquad \delta_a(b) = \emptyset,\ \delta_a(\$) = \mathsf{Accept}, \\ M_b : \delta_b(¢) = \mathsf{MVR},\ \delta_b(a) = \mathsf{MVR},\ \delta_b(b) = \varepsilon,\ \delta_b(\$) = \emptyset. \end{array}$$

Thus, we have the following incomparability result.

COROLLARY 4.9 $\mathcal{L}_{=1}(\text{stl-det-strict-CD-R(1)})$ *and* $\mathcal{L}_{=1}(\text{stl-det-global-CD-R(1)})$ *are incomparable to the class of rational trace languages with respect to inclusion.*

Next we study some closure and nonclosure properties of the language class $\mathcal{L}_{=1}(\text{stl-det-global-CD-R(1)})$.

4.1 *Closure and Nonclosure Properties of* $\mathcal{L}_{=1}$(stl-det-global-CD-R(1))

We first look at the Boolean operations, morphisms, and the commutative closure.

PROPOSITION 4.10

(a) $\mathcal{L}_{=1}$(stl-det-global-CD-R(1)) *is closed under complementation.*
(b) $\mathcal{L}_{=1}$(stl-det-global-CD-R(1)) *is not closed under union, intersection with regular sets, and alphabetic morphisms.*

Proof. (a) Assume that the language $L \subseteq \Sigma^*$ is accepted by a stl-det-global-CD-R(1)-system $\mathcal{M} = ((M_i, \sigma_i)_{i\in I}, I_0, \delta)$ in normal form, and let $\mathcal{M}^c$ be the system that is obtained from $\mathcal{M}$ by changing Accept transitions into undefined transitions and vice versa. Then $L_{=1}(\mathcal{M}^c) = \Sigma^* \setminus L_{=1}(\mathcal{M}) = \Sigma^* \setminus L$, which shows that $\mathcal{L}_{=1}$(stl-det-global-CD-R(1)) is indeed closed under complementation.

(b) Obviously,

$$L_\vee = \{\, w \in \{a,b\}^* \mid |w|_a = |w|_b \geq 0 \,\} \cup \{\, w \in \{a,b\}^* \mid 2 \cdot |w|_a = |w|_b \geq 0 \,\}.$$

As the languages $\{\, w \in \{a,b\}^* \mid |w|_a = |w|_b \geq 0 \,\}$ and $\{\, w \in \{a,b\}^* \mid 2 \cdot |w|_a = |w|_b \geq 0 \,\}$ are both accepted by stl-det-global-CD-R(1))-systems, while $L_\vee$ is not by Proposition 4.7, it follows that $\mathcal{L}_{=1}$(stl-det-global-CD-R(1)) is not closed under union.

The language $\{\, a^n b^n \mid n \geq 0 \,\} = \{\, w \in \{a,b\}^* \mid |w|_a = |w|_b \geq 0 \,\} \cap (a^* \cdot b^*)$ does not contain a regular sublanguage that is letter-equivalent to the language itself. Hence, this language is not even accepted by any stl-det-local-CD-R(1)-system by Proposition 2.2 (a). Thus, $\mathcal{L}_{=1}$(stl-det-global-CD-R(1)) is not closed under intersection with regular sets.

Finally, let $\Gamma = \{a,b,c,d\}$. Then the language

$$L'_\vee = \{\, w \in \{a,b\}^* \mid |w|_a = |w|_b \geq 0 \,\} \cup \{\, w \in \{c,d\}^* \mid 2 \cdot |w|_c = |w|_d \geq 0 \,\}$$

is accepted by a stl-det-global-CD-R(1)-system. Define an alphabetic morphism $h : \Gamma^* \to \{a,b\}^*$ through $a \mapsto a$, $b \mapsto b$, $c \mapsto a$, and $d \mapsto b$. Then $h(L'_\vee) = L_\vee$. It follows that $\mathcal{L}_{=1}$(stl-det-global-CD-R(1)) is not closed under alphabetic morphisms. □

PROPOSITION 4.11 *The language class* $\mathcal{L}_{=1}$(stl-det-global-CD-R(1)) *is not closed under the operation of taking the commutative closure.*

Proof. $L_\vee$ is the commutative closure of the regular language $(ab)^* \cup (abb)^*$. Hence, Lemma 4.3 and Proposition 4.7 yield the stated nonclosure property. □

Recall from [11] that the language class $\mathcal{L}_{=1}$(stl-det-local-CD-R(1)) is closed under the operation of taking the commutative closure. Next we study the operations of product, Kleene star, and reversal.

Let $\Sigma_0 = \{a,b\}$, and let $L_\geq = \{\, u \in \Sigma_0^* \mid |u|_a \geq |u|_b \geq 0 \,\}$. For this language we have the following technical results.

LEMMA 4.12

(a) $L_\geq \in \mathcal{L}_{=1}$(stl-det-global-CD-R(1)).
(b) $L_\geq$ *is not accepted by any* stl-det-global-CD-R(1)*-system that first completely erases the given input and that then accepts on the empty word.*

Proof. (a) Let $\mathcal{M} = ((M_i, \sigma_i)_{i\in I}, I_0, \delta)$ be the stl-det-global-CD-R(1)-system that

is defined by taking $I = \{0, 1, 2\}$, $I_0 = \{0\}$, and $\sigma_0 = \{1, 2\}$, $\sigma_1 = \{0\} = \sigma_2$, by defining the automata M_0, M_1, M_2 through

$$\begin{array}{llll} M_0 : \delta_0(¢) = \mathsf{MVR}, & \delta_0(a) = \varepsilon, & \delta_0(b) = \varepsilon, & \delta_0(\$) = \mathsf{Accept}, \\ M_1 : \delta_1(¢) = \mathsf{MVR}, & \delta_1(a) = \mathsf{MVR}, & \delta_1(b) = \varepsilon, & \delta_1(\$) = \mathsf{Accept}, \\ M_2 : \delta_2(¢) = \mathsf{MVR}, & \delta_2(a) = \varepsilon, & \delta_2(b) = \mathsf{MVR}, & \delta_2(\$) = \emptyset, \end{array}$$

and by defining the successor function δ through $\delta(0, a) = 1$, $\delta(0, b) = 2$, $\delta(1, b) = 0$, and $\delta(2, a) = 0$.

Given a word $u \in \{a, b\}^*$ as input, the system $\mathcal{M}$ proceeds as follows. If $u = \varepsilon$, then the initial component M_0 performs a single move-right step and then accepts; otherwise, it deletes the first symbol s of u. If $s = a$, then M_1 becomes active. It deletes the leftmost occurrence of the symbol b, if there is any, otherwise it simply accepts on reaching the \$-symbol. If $s = b$, then M_2 becomes active, which deletes the leftmost occurrence of the symbol a, if there is any, otherwise it gets stuck on reaching the \$-symbol. In both cases, if a symbol is deleted, then M_0 becomes active again. In this case one occurrence of a and of b each has been deleted. It now follows easily from this description that $L_{=1}(\mathcal{M}) = L_{\geq}$, that is, $L_{\geq} \in \mathcal{L}_{=1}(\mathsf{stl\text{-}det\text{-}global\text{-}CD\text{-}R(1)})$.

(b) Let $\mathcal{M}' = ((M_i, \sigma_i)_{i \in I}, \{i_0\}, \delta)$ be a stl-det-global-CD-R(1)-system accepting $L_{\geq}$ such that each accepting computation of $\mathcal{M}'$ is of the form $(i_0, u) \vdash^{c^{|u|}}_{\mathcal{M}'} (j, \varepsilon) \vdash^2_{M_j} \mathsf{Accept}$, that is, during the first $|u|$ many cycles the word u is completely erased, and then the component M_j reached accepts in two steps starting with the tape contents ¢\$. We claim that $L_{=1}(\mathcal{M}') \neq L_{\geq}$.

Assume that $L_{\geq} \subseteq L_{=1}(\mathcal{M}')$. Let $u = a^n$, where $n > |I|$. Then $u \in L_{\geq}$, and hence $\mathcal{M}'$ accepts on input u. According to our assumption above, the accepting computation of $\mathcal{M}'$ on input u has the following form, where $i_j \in I$ for all $j = 1, \ldots, n$:

$$(i_0, u) = (i_0, a^n) \vdash^c_{\mathcal{M}'} (i_1, a^{n-1}) \vdash^c_{\mathcal{M}'} \cdots \vdash^c_{\mathcal{M}'} (i_n, \varepsilon) \vdash^2_{M_{i_n}} \mathsf{Accept}.$$

As $n > |I|$, there are indices r, s, $0 \leq r < s \leq n$, such that $i_r = i_s$.

Now consider input $z = a^n b^n \in L_{\geq}$. As $\mathcal{M}'$ is globally deterministic, the computation of $\mathcal{M}'$ on input z has the following form:

$$(i_0, z) \vdash^c_{\mathcal{M}'} (i_1, a^{n-1}b^n) \vdash^c_{\mathcal{M}'} \cdots \vdash^c_{\mathcal{M}'} (i_n, b^n) \vdash^{c^n}_{\mathcal{M}'} (i_m, \varepsilon) \vdash^2_{M_m} \mathsf{Accept}$$

for some $i_m \in I$. As $i_r = i_s$, $\mathcal{M}'$ will perform the following computation on input $a^{n-s+r}b^n$:

$$\begin{array}{rcl} (i_0, a^{n-s+r}b^n) \vdash^{c^r}_{\mathcal{M}'} (i_r, a^{n-s}b^n) & = & (i_s, a^{n-s}b^n) \vdash^{c^{n-s}}_{\mathcal{M}'} \cdots \\ \vdash^c_{\mathcal{M}'} (i_n, b^n) & \vdash^{c^n}_{\mathcal{M}'} & (i_m, \varepsilon) \vdash^2_{M_m} \mathsf{Accept}. \end{array}$$

Since $n - s + r \leq n - 1$, we have $a^{n-s+r}b^n \notin L_{\geq}$, which implies that $L_{\geq}$ is in fact a proper subset of $L_{=1}(\mathcal{M}')$. This completes the proof of (b). □

Lemma 4.12 implies in particular that for stl-det-global-CD-R(1)-systems we do not have the *strong normal form* that we have for stl-det-local-CD-R(1)-systems (see, [12]). Based on this technical lemma we can now prove the following nonclosure property.

COROLLARY 4.13 *The language class* $\mathcal{L}_{=1}(\mathsf{stl\text{-}det\text{-}global\text{-}CD\text{-}R(1)})$ *is not closed un-*

der product.

Proof. We consider the product of the languages $L_{\geq}$ and $L_c = \{c\}$, where $c \notin \{a, b\}$ is a new letter. While $L_{\geq} \in \mathcal{L}_{=1}(\mathsf{stl\text{-}det\text{-}global\text{-}CD\text{-}R(1)})$ by Lemma 4.12 (a), L_c is regular, and so it is accepted by a stl-det-global-CD-R(1)-system by Lemma 4.3. We claim, however, that the language

$$L_{pr} = L_{\geq} \cdot L_c = \{\, uc \mid u \in \{a, b\}^*,\ |u|_a \geq |u|_b \geq 0 \,\}$$

is not accepted by any stl-det-global-CD-R(1)-system.

Assume to the contrary that $\mathcal{M} = ((M_i, \sigma_i)_{i \in I}, \{i_0\}, \delta)$ is a stl-det-global-CD-R(1)-system such that $L_{=1}(\mathcal{M}) = L_{pr}$. To derive the intended contradiction we need the following claim.

Claim. For each word $w = uc \in L_{pr}$, the accepting computation of $\mathcal{M}$ on input w must be of the form $(i_0, w) = (i_0, uc) \vdash_{\mathcal{M}}^{c^{|u|}} (i_m, c) \vdash_{\mathcal{M}}^{c} (j, \varepsilon) \vdash_{\mathcal{M}}^{2} \mathsf{Accept}$.

Proof. As $\mathcal{M}$ must verify that the given input w ends with the symbol c, one of the components of $\mathcal{M}$ must read the symbol c in the course of the accepting computation of $\mathcal{M}$ on input w. Assume that M_{i_m} is this particular component. If $\delta_{i_m}(c)$ is undefined, then M_{i_m} would get stuck, and so the computation of $\mathcal{M}$ on input w would not accept. Thus, $\delta_{i_m}(c)$ is defined.

If $\delta_{i_m}(c) = \mathsf{MVR}$, then after executing the corresponding step, M_{i_m} would read the \$-symbol. As the computation considered is accepting, this means that M_{i_m} must accept at this point. But then $\mathcal{M}$ would also accept the word $wc = ucc \notin L_{pr}$, as the computation of $\mathcal{M}$ on input wc would be exactly the same as the one on input w. This, however, contradicts our assumption above. Hence, it follows that $\delta_{i_m}(c) = \varepsilon$.

Let $j = \delta(i_m, c)$ be the index of the corresponding successor component. Then the accepting computation of $\mathcal{M}$ on input w has the form

$$(i_0, w) = (i_0, uc) \vdash_{\mathcal{M}}^{r} (i_m, vc) \vdash_{\mathcal{M}}^{c} (j, v) \vdash_{\mathcal{M}}^{*} \mathsf{Accept}.$$

Thus, it remains to show that $v = \varepsilon$, that is, $r = |u|$. Assume to the contrary that $v \neq \varepsilon$. Then $\delta_{i_m}(x) = \mathsf{MVR}$ for all letters $x \in \{a, b\}$ satisfying $|v|_x \geq 1$. Let $v = v'x$, where $x \in \{a, b\}$, and let $z = u'cx$ be the word that is obtained from $w = uc$ by moving the last occurrence of the letter x to the right end of the word. Then the computation of $\mathcal{M}$ on input z looks as follows:

$$(i_0, z) = (i_0, u'cx) \vdash_{\mathcal{M}}^{r} (i_m, v'cx) \vdash_{\mathcal{M}}^{c} (j, v'x) = (j, v) \vdash_{\mathcal{M}}^{*} \mathsf{Accept}.$$

This, however, contradicts our assumption above, as $z = u'cx \notin L_{pr}$. Hence, it follows that $v = \varepsilon$, which proves the above claim. □

Continuing with the proof of Corollary 4.13, we note that, for each component M_i that can only encounter an occurrence of the symbol c in a non-accepting computation, one can simply take $\delta_i(c)$ to be undefined.

Now we modify the system $\mathcal{M}$ to obtain a stl-det-global-CD-R(1)-system $\mathcal{M}'$ as follows. For each index $i \in I$, if $\delta_i(c)$ is defined, that is, if $\delta_i(c) = \varepsilon$ according to our observations above, then we remove this transition and take $\delta_i(\$) = \mathsf{Accept}$. Then, for each word $u \in L_{\geq}$, the computation of $\mathcal{M}'$ on input u will parallel the computation of $\mathcal{M}$ on input uc, and thus, we see from the claim above that it will first erase u completely and then accept on reaching the empty word. Now the

proof of Lemma 4.12 (b) implies that $L_{\geq} \subsetneq L_{=1}(\mathcal{M}')$, that is, there exists a word $u \in \{a,b\}^* \setminus L_{\geq}$ such that $\mathcal{M}'$ accepts on input u. But then $\mathcal{M}$ will accept on input $uc \notin L_{pr}$, which contradicts our assumption above. Hence, it follows that L_{pr} is not accepted by any stl-det-global-CD-R(1)-system. □

Consider the language $L_{pr}^R = \{ cu \mid u \in \{a,b\}^*, |u|_a \geq |u|_b \geq 0 \}$. From Lemma 4.12 (a) we have a stl-det-global-CD-R(1)-system $\mathcal{M} = ((M_i, \sigma_i)_{i \in I}, \{0\}, \delta)$ for accepting the language $L_{\geq}$. Let $\mathcal{M}'$ be obtained from $\mathcal{M}$ by introducing a new initial component M_{ini} that is described by the transition function δ_{ini} defined by $\delta_{ini}(¢) = \mathsf{MVR}$, $\delta_{ini}(c) = \varepsilon$, and $\delta_{ini}(a) = \delta_{ini}(b) = \delta_{ini}(\$) = \emptyset$, and the successor set $\sigma_{ini} = \{0\}$, and by extending the successor function δ by taking $\delta(ini, c) = 0$. Then it is easily seen that $L_{=1}(\mathcal{M}') = L_{pr}^R$ holds. Thus, together with the fact that the language L_{pr} is not accepted by any stl-det-global-CD-R(1)-system this yields the following additional nonclosure result.

COROLLARY 4.14 *The language class* $\mathcal{L}_{=1}$(stl-det-global-CD-R(1)) *is not closed under reversal.*

Finally we want to prove that the language class $\mathcal{L}_{=1}$(stl-det-global-CD-R(1)) is not closed under Kleene plus. For doing so, we introduce the following variant of the language L_{pr}:

$$L_{pra} = L_{pr} \cdot \{a\}^* = \{ uca^n \mid u \in \{a,b\}^*, |u|_a \geq |u|_b \geq 0, n \geq 0 \}.$$

LEMMA 4.15 $L_{pra} \in \mathcal{L}_{=1}$(stl-det-global-CD-R(1)).

Proof. Let $\mathcal{M} = ((M_i, \sigma_i)_{i \in I}, \{0\}, \delta)$ be the stl-det-global-CD-R(1)-system that is defined by taking $I = \{0,1,2,3\}$, $\sigma_0 = \{1,2,3\}$, $\sigma_1 = \{0,3\}$, $\sigma_2 = \sigma_3 = \{0\}$, by defining the automata M_0, M_1, M_2, M_3 through

$$\begin{array}{llllll}
M_0 : & \delta_0(¢) = \mathsf{MVR}, & \delta_0(a) = \varepsilon, & \delta_0(b) = \varepsilon, & \delta_0(c) = \varepsilon, & \delta_0(\$) = \emptyset, \\
M_1 : & \delta_1(¢) = \mathsf{MVR}, & \delta_1(a) = \mathsf{MVR}, & \delta_1(b) = \varepsilon, & \delta_1(c) = \varepsilon, & \delta_1(\$) = \emptyset, \\
M_2 : & \delta_2(¢) = \mathsf{MVR}, & \delta_2(a) = \varepsilon, & \delta_2(b) = \mathsf{MVR}, & \delta_2(c) = \emptyset, & \delta_2(\$) = \emptyset, \\
M_3 : & \delta_3(¢) = \mathsf{MVR}, & \delta_3(a) = \mathsf{MVR}, & \delta_3(b) = \emptyset, & \delta_3(c) = \emptyset, & \delta_3(\$) = \mathsf{Accept},
\end{array}$$

and the successor function δ is defined through $\delta(0,a) = 1$, $\delta(0,b) = 2$, $\delta(0,c) = 3$, $\delta(1,b) = 0$, $\delta(1,c) = 3$, and $\delta(2,a) = 0$.

The transition function δ together with the component M_3 ensure that each word accepted contains a single occurrence of the letter c, that is, it is of the form $w = ucv$ for some $u, v \in \{a,b\}^*$. Now the components M_0, M_1, and M_2 together verify that $|u|_a \geq |u|_b \geq 0$ holds, that is, that $u \in L_{\geq}$, and the component M_3 simply accepts if $v \in \{a\}^*$. Hence, it follows that $L_{=1}(\mathcal{M}) = L_{pra}$. □

While $L_{pra} \in \mathcal{L}_{=1}$(stl-det-global-CD-R(1)), we have the following negative result on the language $L_+ = (L_{pra})^+$.

LEMMA 4.16 $L_+ = (L_{pra})^+ \notin \mathcal{L}_{=1}$(stl-det-global-CD-R(1)).

Proof. Let $\mathcal{M} = ((M_i, \sigma_i)_{i \in I}, I_0, \delta)$ be a stl-det-global-CD-R(1)-system such that $L_{=1}(\mathcal{M})$ contains the language

$$L_{pr} \cdot L_{pra} = \{ ucvca^m \mid u, v \in \{a,b\}^*, |u|_a \geq |u|_b \geq 0, |v|_a \geq |v|_b \geq 0, m \geq 0 \}.$$

By Proposition 4.5 we can assume that $\mathcal{M}$ is in normal form.

Claim. $L_{=1}(\mathcal{M}) \not\subseteq L_+$.

Proof. Assume to the contrary that $L_{pr} \cdot L_{pra} \subseteq L_{=1}(\mathcal{M}) \subseteq L_+$ holds. From this assumption we will derive a contradiction.

We consider the computations of $\mathcal{M}$ on inputs of the form $w = a^{n_1}b^{n_2}ca^{n_3}b^{n_4}c$, where $n_1, n_2, n_3, n_4 > |I|$ are large positive integers. If $n_1 \geq n_2$ and $n_3 \geq n_4$, then $w \in L_{pr} \cdot L_{pra}$, and we see from our assumption above that the corresponding computation is accepting, that is, it is of the form

$$(i_0, w) \vdash^c_{\mathcal{M}} (i_1, w_1) \vdash^c_{\mathcal{M}} \cdots \vdash^c_{\mathcal{M}} (i_k, w_k) \vdash^*_{M_{i_k}} \mathsf{Accept},$$

where M_{i_0} is the initial component of $\mathcal{M}$, and the last part $(i_k, w_k) \vdash^*_{M_{i_k}} \mathsf{Accept}$ is an accepting tail computation. If $k = 0$, that is, if already the initial component performs an accepting tail computation, then together with w, $\mathcal{M}$ would also accept the word $z = a^{n_1}b^{n_2}cb^{n_3} \notin L_+$. Hence, we see that $k \geq 1$, and that M_{i_0} executes a delete operation on w. Also, as $\mathcal{M}$ is in normal form, $\delta_{i_0}(x) \in \{\mathsf{MVR}, \varepsilon\}$ for all $x \in \{a, b, c\}$.

We now consider several cases:

- If $\delta_{i_0}(a) = \mathsf{MVR}$ and $\delta_{i_0}(c) = \varepsilon$, then, for each $n \geq 1$, $\mathcal{M}$ would perform the cycles $(i_0, a^nca^{n+1}b^{n+1}c) \vdash^c_{\mathcal{M}} (j, a^{2n+1}b^{n+1}c)$ and $(i_0, a^{n+1}ca^nb^{n+1}c) \vdash^c_{\mathcal{M}} (j, a^{2n+1}b^{n+1}c)$ for some $j \in I$. As $a^nca^{n+1}b^{n+1}c \in L_{pr} \cdot L_{pra}$, we see that the computation starting from the restarting configuration $(j, a^{2n+1}b^{n+1}c)$ is accepting. This, however, implies that also the word $a^{n+1}ca^nb^{n+1}c \notin L_+$ is accepted.

- If $\delta_{i_0}(a) = \mathsf{MVR} = \delta_{i_0}(c)$ and $\delta_{i_0}(b) = \varepsilon$, then, for each $n \geq 1$, $\mathcal{M}$ would perform the cycles $(i_0, a^nbca^nb^nc) \vdash^c_{\mathcal{M}} (j, a^nca^nb^nc)$ and $(i_0, a^ncba^nb^nc) \vdash^c_{\mathcal{M}} (j, a^nca^nb^nc)$ for some $j \in I$. As $a^nbca^nb^nc \in L_{pr} \cdot L_{pra}$, we see that the computation starting from the restarting configuration $(j, a^nca^nb^nc)$ is accepting. This, however, implies that also the word $a^ncba^nb^nc \notin L_+$ is accepted.

It follows that $\delta_{i_0}(a) = \varepsilon$. Thus, the accepting computation of $\mathcal{M}$ on an input of the form a^nb^mcvc, $n \geq m > |I|$ and $v \in L_{\geq}$, begins with a finite sequence of cycles in each of which the first occurrence of the letter a is deleted, that is, we can factor it as

$$(i_0, a^nb^mcvc) \vdash^{c^r}_{\mathcal{M}} (j_r, a^{n-r}b^mcvc) \vdash^c_{\mathcal{M}} (j_{r+1}, w_{r+1}) \vdash^*_{\mathcal{M}} \mathsf{Accept},$$

where the component j_r will not erase an occurrence of the letter a, that is, $\delta_{j_r}(a) = \mathsf{MVR}$. Observe that we have $r \leq |I|$, since otherwise some component would occur repeatedly in this initial sequence of cycles, and we could use pumping to accept a word of the form $a^{n-s}b^ncvc \notin L_+$ for some integer s satisfying $0 < s < n$ together with the word $a^nb^ncvc \in L_{pr} \cdot L_{pra}$.

We now analyse the behaviour of M_{j_r}.

- Assume that $\delta_{j_r}(b) = \mathsf{MVR}$. If $\delta_{j_r}(c) = \mathsf{MVR}$, then M_{j_r} would only perform tail computations. Hence, $\delta_{i_r}(\$) = \mathsf{Accept}$, as we are considering an accepting computation. This, however, would again imply that $\mathcal{M}$ would accept on input $a^nb^{2n}cvc \notin L_+$. Finally, if $\delta_{j_r}(c) = \varepsilon$, then $w_{r+1} = a^{n-r}b^mvc$ would follow, implying that

$$(i_0, a^nb^mcvc) \vdash^{c^r}_{\mathcal{M}} (j_r, a^{n-r}b^mcvc) \vdash^c_{\mathcal{M}} (j_{r+1}, a^{n-r}b^mvc) \vdash^*_{\mathcal{M}} \mathsf{Accept}$$

is an accepting computation of $\mathcal{M}$. For $m = 2n$ this implies that $\mathcal{M}$ accepts on input $a^n b^{2n} cvc \notin L_+$. Hence, it follows that $\delta_{j_r}(b) = \varepsilon$.

- If $\delta_{j_r}(c) = \mathsf{MVR}$, then $\mathcal{M}$ would perform the computations

$$(i_0, a^n bca^n b^n c) \vdash_{\mathcal{M}}^{c^r} (j_r, a^{n-r} bca^n b^n c) \vdash_{\mathcal{M}}^{c} (j_{r+1}, a^{n-r} ca^n b^n c)$$

and

$$(i_0, a^n cba^n b^n c) \vdash_{\mathcal{M}}^{c^r} (j_r, a^{n-r} cba^n b^n c) \vdash_{\mathcal{M}}^{c} (j_{r+1}, a^{n-r} ca^n b^n c).$$

As $a^n bca^n b^n c \in L_{pr} \cdot L_{pra}$, $(j_{r+1}, a^{n-r} ca^n b^n c) \vdash_{\mathcal{M}}^{*} \mathsf{Accept}$, which implies that also the word $a^n cba^n b^n c \notin L_+$ is accepted by $\mathcal{M}$. Finally, if $\delta_{j_r}(c) = \varepsilon$, then $\mathcal{M}$ would perform the computations

$$(i_0, a^n ca^n b^n c) \vdash_{\mathcal{M}}^{c^r} (j_r, a^{n-r} ca^n b^n c) \vdash_{\mathcal{M}}^{c} (j_{r+1}, a^{2n-r} b^n c)$$

and

$$(i_0, a^{2n} cb^n c) \vdash_{\mathcal{M}}^{c^r} (j_r, a^{2n-r} cb^n c) \vdash_{\mathcal{M}}^{c} (j_{r+1}, a^{2n-r} b^n c).$$

As $a^n ca^n b^n c \in L_{pr} \cdot L_{pra}$, $(j_{r+1}, a^{2n-r} b^n c) \vdash_{\mathcal{M}}^{*} \mathsf{Accept}$, which implies that also the word $a^{2n} cb^n c \notin L_+$ is accepted by $\mathcal{M}$.

As this covers all cases we conclude that indeed $L_{=1}(\mathcal{M}) \not\subseteq L_+$ holds. □

Since $L_{pr} \cdot L_{pra} \subseteq L_+$, the above claim shows that the language L_+ is not accepted by any stl-det-global-CD-R(1)-system. □

From this lemma and its proof we now obtain the following nonclosure results.

COROLLARY 4.17 *The language class* $\mathcal{L}_{=1}$(stl-det-global-CD-R(1)) *is neither closed under Kleene plus nor under Kleene star.*

The following table summarizes the closure and nonclosure properties of the language classes that are accepted by the various types of stateless CD-R(1)-systems:

Type of CD-System	**Operations**								
	$\cup$	$\cap_{REG}$	^{c}	$\cdot$	$^+$	h	h^{-1}	com	^{R}
stl-det-local-CD-R(1)	+	−	−	+	+	−	?	+	?
stl-det-global-CD-R(1)	−	−	+	−	−	−	?	−	−
stl-det-strict-CD-R(1)	−	−	+	−	−	−	−	−	−

Here the operations are abbreviated as follows:

- $\cup$ denotes the operation of union,
- $\cap_{REG}$ denotes the intersection with a regular language,
- ^{c} denotes the operation of complementation,
- $\cdot$ denotes the product operation,
- $^+$ denotes the Kleene plus,
- h denotes the application of an alphabetic morphism,
- h^{-1} denotes the operation of taking the preimage with respect to a morphism,
- com denotes the operation of taking the commutative closure,
- ^{R} denotes the operation of taking the reversal,

and "+" denotes the fact that the corresponding class is closed under the given operation, "−" denotes the fact that it is not closed, and "?" indicates that the status of this property is still open.

Finally we look at the closure of the language class $\mathcal{L}_{=1}$(stl-det-global-CD-R(1)) with respect to the operations of intersection with regular sets and projections. 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 $L_\Sigma := \{ \mathrm{sh}(w, \overline{w}) \mid w \in \Sigma^* \}$, where $\mathrm{sh}(w, \overline{w})$ denotes the *shuffle* of the two words w and $\overline{w}$, is called the *twin shuffle language* over Σ. Further, let $\mathsf{Pr}^{\Sigma_T} : (\Sigma \cup \overline{\Sigma})^* \to \Sigma_T^*$ denote the projection from $(\Sigma \cup \overline{\Sigma})^*$ onto Σ_T^* for a subalphabet Σ_T of Σ. As shown by the following classical result, the twin shuffle languages are quite expressive

PROPOSITION 4.18 [15] *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 one can easily design a stl-det-global-CD-R(1)-system $\mathcal{M}_\Sigma$ such that $L_{=1}(\mathcal{M}_\Sigma) = L_\Sigma$. Hence, we obtain the following consequence.

COROLLARY 4.19 *For each recursively enumerable language $L \subseteq \Sigma_T^*$, there are an alphabet Σ containing Σ_T, a language $L_1 \in \mathcal{L}_{=1}$(stl-det-global-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, the closure of the language class $\mathcal{L}_{=1}$(stl-det-global-CD-R(1)) under intersection with regular sets and projections already yields all recursively enumerable languages.

5. Decision Problems

Finally we take a look at some standard decision problems for stl-det-global-CD-R(1)-systems. As these systems are a special type of stl-det-local-CD-R(1)-systems, we inherit the following decidability results from [12].

COROLLARY 5.1 *The membership problem, the emptiness problem, and the finiteness problem are effectively decidable for* stl-det-global-CD-R(1)*-systems.*

By Proposition 4.10 (a) the language class $\mathcal{L}_{=1}$(stl-det-global-CD-R(1)) is (effectively) closed under the operation of complementation. Thus, we obtain the following from the decidability of the emptiness problem.

COROLLARY 5.2 *The universe problem is effectively decidable for* stl-det-global-CD-R(1)*-systems, that is, it is decidable whether $L_{=1}(\mathcal{M}) = \Sigma^*$ for a given* stl-det-

global-CD-R(1)-*system* $\mathcal{M}$ *on* Σ.

In [12] it is shown that the regularity, the inclusion and the equivalence problems are undecidable for stl-det-local-CD-R(1)-systems. The proofs for these undecidability results rest on the fact that the universe problem is undecidable for stl-det-local-CD-R(1)-systems. Thus, this proof does not carry over to stl-det-global-CD-R(1)-systems. Accordingly we have to find a new approach if we want to establish corresponding undecidability results for stl-det-global-CD-R(1)-systems.

Below we begin this investigation by studying the following variant of the intersection emptiness problem:

Intersection With Regular Language Emptiness Problem:

Instance : A stl-det-global-CD-R(1)-system $\mathcal{M}$ and a finite-state acceptor A.
Question : Does $L_{=1}(\mathcal{M}) \cap L(A) = \emptyset$ hold?

Since all regular languages are accepted by stl-det-global-CD-R(1)-systems (Lemma 4.3), this is indeed a special variant of the intersection emptiness problem for stl-det-global-CD-R(1)-systems. As already indicated by Corollary 4.19, the languages of the form $L_{=1}(\mathcal{M}) \cap L(A)$ are quite complex. Hence, the following undecidability result does not come as a surprize.

THEOREM 5.3 *The Intersection With Regular Language Emptiness Problem is undecidable for* stl-det-global-CD-R(1)*-systems.*

Proof. We prove the undecidability of this problem by a reduction from the *Post Correspondence Problem* (PCP), which can be stated as follows (see, e.g., [4]):

Instance : Two morphisms $f, g : \Sigma^* \to \Delta^*$.
Question : Is there a non-empty word $w \in \Sigma^+$ such that $f(w) = g(w)$?

It is well-known that the PCP is undecidable in general. Let $f, g : \Sigma^* \to \Delta^*$ be two morphisms, where we can assume without loss of generality that the two alphabets Σ and Δ are disjoint. With each of the morphisms f and g we now associate a language; however, the languages L_f associated with f and L_g associated with g are defined differently:

$$L_f = \{\, \mathrm{sh}(w, f(w)) \mid w \in \Sigma^+ \,\} \cdot \#, \text{ and } L_g = \{\, ag(a) \mid a \in \Sigma \,\}^+ \cdot \#.$$

Here $\#$ is a new symbol, and as mentioned before $\mathrm{sh}(u, v)$ denotes the *shuffle* of u and v. Obviously, the language L_g is regular, and from g we can easily construct a finite-state acceptor A_g for this language.

Claim 1. $L_f \in \mathcal{L}_{=1}$(stl-det-global-CD-R(1)).

Proof. Let $\mathcal{M}_f = ((M_i, \sigma_i)_{i \in I}, \{0\}, \delta)$ be the stl-det-global-CD-R(1)-system on $\Gamma = \Sigma \cup \Delta \cup \{\#\}$ that is defined as follows:

- $I' = \{\, (f(a), i) \mid a \in \Sigma,\ 1 \leq i \leq |f(a)| \,\}$ and $I = \{0, 1, +\} \cup I'$,
- the successor sets are defined through

$$\begin{array}{lll} \sigma_0 = I' \cup \{1\}, & \sigma_{(f(a),i)} = \{(f(a), i+1)\} & \text{for all } a \in \Sigma \text{ and all } 1 \leq i < |f(a)|, \\ \sigma_1 = I' \cup \{1, +\}, & \sigma_{(f(a),i)} = \{1\} & \text{for all } a \in \Sigma \text{ and } i = |f(a)|, \\ \sigma_+ = \{+\}, & & \end{array}$$

- the automata M_0, M_1, and M_+ are defined through

$$\begin{array}{lllll} \delta_0(¢) = \mathsf{MVR}, & \delta_1(¢) = \mathsf{MVR}, & \delta_+(¢) = \mathsf{MVR}, & & \\ \delta_0(a) = \varepsilon, & \delta_1(a) = \varepsilon, & \delta_+(a) = \emptyset & \text{for all } a \in \Sigma, \\ \delta_0(b) = \mathsf{MVR}, & \delta_1(b) = \mathsf{MVR}, & \delta_+(b) = \emptyset & \text{for all } b \in \Delta, \\ \delta_0(\#) = \emptyset, & \delta_1(\#) = \varepsilon, & \delta_+(\#) = \emptyset, & \\ \delta_0(\$) = \emptyset, & \delta_1(\$) = \emptyset, & \delta_+(\$) = \mathsf{Accept}, & \end{array}$$

- for all $a \in \Sigma$ and all $1 \leq i \leq |f(a)|$, the automaton $M_{(f(a),i)}$ is defined as follows, where $f(a) = b_1 \dots b_m$, $m \geq 1$, $b_1 \dots, b_m \in \Delta$,

$$\begin{array}{lll} \delta_{(f(a),i)}(¢) = \mathsf{MVR}, & & \delta_{(f(a),i)}(b_i) = \varepsilon, \\ \delta_{(f(a),i)}(a) = \mathsf{MVR} & \text{for all } a \in \Sigma, & \delta_{(f(a),i)}(\#) = \emptyset, \\ \delta_{(f(a),i)}(b) = \emptyset & \text{for all } b \in \Delta \setminus \{b_i\}, & \delta_{(f(a),i)}(\$) = \emptyset, \end{array}$$

- and the successor function δ is defined through

$$\begin{array}{rl} \delta(0,a) = 1 & \text{for all } a \in \Sigma \text{ satisfying } f(a) = \varepsilon, \\ \delta(0,a) = (f(a),1) & \text{for all } a \in \Sigma \text{ satisfying } f(a) \neq \varepsilon, \\ \delta(1,a) = 1 & \text{for all } a \in \Sigma \text{ satisfying } f(a) = \varepsilon, \\ \delta(1,a) = (f(a),1) & \text{for all } a \in \Sigma \text{ satisfying } f(a) \neq \varepsilon, \\ \delta(1,\#) = +, & \\ \delta((f(a),i),b_i) = (f(a),i+1), & \text{if } f(a) = b_1 \dots b_m \text{ and } 1 \leq i < m = |f(a)|, \\ \delta((f(a),i),b_i) = 1, & \text{if } f(a) = b_1 \dots b_m \text{ and } 1 \leq i = m = |f(a)|. \end{array}$$

Then it is quite easily verified that $L_{=1}(\mathcal{M}_f) = L_f$ holds. □

There exists a non-empty word $w \in \Sigma^+$ such that $f(w) = g(w)$, if and only if there exists a word $w = a_{i_1}a_{i_2}\dots a_{i_r} \in \Sigma^+$ ($r \geq 1$, $a_{i_1}, \dots, a_{i_r} \in \Sigma$) such that $a_{i_1}g(a_{i_1})a_{i_2}g(a_{i_2})\dots a_{i_r}g(a_{i_r}) \in \mathrm{sh}(a_{i_1}a_{i_2}\dots a_{i_r}, f(a_{i_1}a_{i_2}\dots a_{i_r}))$, if and only if there exists a word $w = a_{i_1}a_{i_2}\dots a_{i_r} \in \Sigma^+$ such that $a_{i_1}g(a_{i_1})a_{i_2}g(a_{i_2})\dots a_{i_r}g(a_{i_r}) \cdot \# \in L_f \cap L_g$, if and only if $L_f \cap L_g \neq \emptyset$.

As $\mathcal{M}_f$ and A_g are effectively constructible from the given morphisms f and g, and as the PCP is undecidable in general, the above equivalence implies that the Intersection With Regular Language Emptiness Problem is undecidable for stl-det-global-CD-R(1)-systems. □

Based on this undecidability result we can now prove that the following variants of the inclusion problem are undecidable, too.

COROLLARY 5.4 *The following inclusion problems are undecidable in general:*

(1) Inclusion In Regular Language Problem:

Instance : *A* stl-det-global-CD-R(1)*-system* $\mathcal{M}$ *and a finite-state acceptor* A.
Question : *Does* $L_{=1}(\mathcal{M}) \subseteq L(A)$ *hold?*

(2) Containing Regular Language Problem:

Instance : *A* stl-det-global-CD-R(1)*-system* $\mathcal{M}$ *and a finite-state acceptor* A.
Question : *Does* $L(A) \subseteq L_{=1}(\mathcal{M})$ *hold?*

Proof. Let $\mathcal{M}$ be a stl-det-global-CD-R(1)-system on Σ, and let A be a finite-state acceptor on Σ. From $\mathcal{M}$ we can construct a stl-det-global-CD-R(1)-system $\mathcal{M}^c$ for the language $\Sigma^* \setminus L_{=1}(\mathcal{M})$, and from A we can construct a finite-state acceptor

A^c for the language $\Sigma^* \setminus L(A)$. Now

$$L_{=1}(\mathcal{M}) \cap L(A) = \emptyset \text{ iff } L_{=1}(\mathcal{M}) \subseteq L(A^c),$$

and

$$L_{=1}(\mathcal{M}) \cap L(A) = \emptyset \text{ iff } L(A) \subseteq L_{=1}(\mathcal{M}^c).$$

Thus, it follows from Theorem 5.3 that the above inclusion problems are undecidable. □

As each regular language is accepted by some stl-det-global-CD-R(1)-system, Corollary 5.4 yields the following undecidablility result.

COROLLARY 5.5 *The inclusion problem is undecidable for* stl-det-global-CD-R(1)-*systems.*

6. Concluding Remarks

The stl-det-local-CD-R(1)-systems correspond to the nondeterministic finite-state acceptors with translucent letters of [13], and the stl-det-global-CD-R(1)-systems correspond to the deterministic finite-state acceptors with translucent letters. In this respect they form quite a natural type of computing device. However, while it is known that the former CD-systems accept all rational trace languages, and the class of languages accepted by them has fairly nice closure properties [11, 12], we have seen here that the class of languages that are accepted by stl-det-global-CD-R(1)-systems is incomparable to the rational trace languages with respect to inclusion, and that it is not closed under most operations of interest in language theory. Thus, from this perspective it is not a nice language class. However, it remains open whether this class is closed under inverse morphisms.

We also studied another, more restricted, deterministic variant of the stl-det-local-CD-R(1)-systems: the stl-det-strict-CD-R(1)-systems. However, these CD-systems are much too weak, as they do not even accept all finite languages, although they do accept some languages that are not even context-free. As it turned out, the three types of CD-systems of stateless deterministic R(1)-automata give a proper 3-level hierarchy.

Finally we have also considered some basic decision problems for stl-det-global-CD-R(1)-systems. In contrast to the situation for stl-det-local-CD-R(1)-systems, the universe problem is decidable for stl-det-global-CD-R(1)-systems. We could nevertheless show that the inclusion problem remains undecidable, but it is still open whether the regularity problem or the equivalence problem are decidable for these systems.

References

[1] E. Csuhaj-Varjú, J. Dassow, J. Kelemen, and G. Păun. *Grammar Systems. A Grammatical Approach to Distribution and Cooperation*, Gordon and Breach, London, 1994.
[2] 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.
[3] V. Diekert and G. Rozenberg (eds.), *The Book of Traces*, World Scientific, Singapore, 1995.
[4] T. Harju and J. Karhumäki. Morphisms. In: G. Rozenberg and A. Salomaa (eds.), *Handbook of Formal Languages*, Vol. 1, Springer, Berlin, 1997, 439–510.

[5] 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.

[6] 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.

[7] M. Kutrib, H. Messerschmidt and F. Otto. On stateless two-pushdown automata and restarting automata. *Int. J. Found. Comput. Sci.* 21 (2010) 781–798.

[8] H. Messerschmidt and F. Otto. Cooperating distributed systems of restarting automata. *Int. J. Found. Comput. Sci.* 18 (2007) 1333–1342.

[9] 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.

[10] H. Messerschmidt and F. Otto. On deterministic CD-systems of restarting automata. *Int. J. Found. Comput. Sci.* 20 (2009) 185–209.

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

[12] B. Nagy and F. Otto. On CD-systems of stateless deterministic R-automata with window size one. *J. Comput. Syst. Sci.* 78 (2012) 780–806.

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

[14] B. Nagy and F. Otto. Globally deterministic CD-systems of stateless R(1)-automata. In: A.H. Dediu, S. Inenaga, and C. Martin-Vide (eds.), *LATA 2011, Proc.*, *Lect. Notes Comput. Sci. 6638*, Springer, Berlin, 2011, 390–401.

[15] A. Salomaa. *Jewels of Formal Language Theory.* Computer Science Press, Rockville, Maryland, 1981.