CD-Systems of Restarting Automata Governed by Explicit Enable and Disable Conditions

Friedrich Otto

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

Abstract. We introduce a new mode of operation for CD-systems of restarting automata by providing explicit enable and disable conditions in the form of regular constraints. We show that, for each CD-system $\mathcal{M}$ of restarting automata and each mode m of operation considered by Messerschmidt and Otto, there exists a CD-system $\mathcal{M}'$ of restarting automata of the same type as $\mathcal{M}$ that, working in the new mode ed, accepts the language $L_{\mathsf{m}}(\mathcal{M})$ that $\mathcal{M}$ accepts in mode m. Further, we will see that in mode ed, a locally deterministic CD-system of restarting automata of type $\mathsf{RR(W)(W)}$ can be simulated by a locally deterministic CD-system of restarting automata of the more restricted type $\mathsf{R(W)(W)}$. This is the first time that a non-monotone type of R-automaton without auxiliary symbols is shown to be as expressive as the corresponding type of RR-automaton.

Keywords: Restarting automaton, CD-system, modes of operation.

1 Introduction

The restarting automaton was introduced by Jančar et. al. as a formal tool to model the *analysis by reduction*, which is a technique used in linguistics to analyze sentences of natural languages [3]. This technique consists in a stepwise simplification of a given sentence in such a way that the correctness or incorrectness of the sentence is not affected. It is applied primarily in languages that have a free word order.

A (one-way) restarting automaton, RRWW-automaton for short, is a device M that consists of a finite-state control, a flexible tape containing a word delimited by sentinels, and a read/write window of a fixed size. This window is moved from left to right until the control decides (nondeterministically) that the content of the window should be rewritten by some *shorter* string. In fact, the new string may contain auxiliary symbols that do not belong to the input alphabet. After a rewrite, M can continue to move its window until it either halts and accepts, or halts and rejects, or restarts, that is, it places its window over the left end of the tape, and reenters the initial state. Thus, each computation of M can be described through a sequence of cycles that is followed by a tail (which is the part of a computation that follows after the last restart step).

Many restricted types of restarting automata have been studied and put into correspondence to more classical classes of formal languages. For a recent survey see [9] or [10]. Also further extensions of the model have been considered. In particular, in [5] cooperating distributed systems (CD-systens) of restarting automata have been introduced, and it has been shown that CD-systems of restarting automata working in mode $= 1$ correspond to the nonforgetting restarting automata of Messerschmidt and Stamer [4, 8]. Also various other modes of operation were introduced in that paper. In the $= j$ mode ($j \geq 2$) the active component automaton is required to execute exactly j cycles, while in the t mode the active component stays active until it cannot apply another meta-instruction anymore. In that situation a successor component takes over and procedes with the computation. Should that component be unable to execute any meta-instruction at all, neither a rewriting one nor an accepting one, then the computation fails. While the former mode is *static*, as the number of cycles executed by the acitve component automaton is always the same, independent of the actual tape contents, the latter mode is *dynamic*, as the number of cycles executed by the active component automaton depends on the tape contents and that component itself.

Here we introduce and study another dynamic mode of operation for CD-systems of restarting automata by associating explicit *enable* and *disable conditions* in the form of regular expressions with each component automaton. A component automaton can only become active if at that moment its enable condition is satisfied by the current tape contents, and it then stays active until its disable condition is satisfied by the (then modified) tape contents. This is motivated by similar modes of operation considered for CD-grammar systems [1, 2]. We study the expressive power of CD-systems of restarting automata working under this mode of operation, which we call ed *mode*. We will see that, for each CD-system $\mathcal{M}$ of restarting automata and each mode m of operation considered in [5], there exists a CD-system $\mathcal{M}'$ of restarting automata of the same type as $\mathcal{M}$ that, working in mode ed, accepts the language $L_{\mathsf{m}}(\mathcal{M})$ that $\mathcal{M}$ accepts in mode m. In fact, the mode $= 1$ and mode t computations of a CD-system of restarting automata can be simulated by mode ed computations of the same CD-system, if appropriate enable and disable conditions are chosen. On the other hand, the mode ed computations of a CD-system of $\mathsf{RR(W)(W)}$-automata can be simulated by mode $= 1$ computations of a modified CD-system of the same type of restarting automata, which proves that CD-systems of $\mathsf{RR(W)(W)}$-automata working in mode $= 1$ and CD-systems of $\mathsf{RR(W)(W)}$-automata working in mode ed have the same expressive power. Actually these results also extend to CD-systems of $\mathsf{RR(W)(W)}$-automata that are locally deterministic or globally deterministic. However, for strictly deterministic CD-systems of $\mathsf{R(R)(W)}$-automata, the expressive power is properly increased by going from mode $= 1$ computations to mode ed computations. Finally, we will see that in mode ed, a locally deterministic CD-system of restarting automata of type $\mathsf{RR(W)(W)}$ can be simulated by a locally deterministic CD-system of restarting automata of the more restricted type $\mathsf{R(W)(W)}$. This is the first time that a non-monotone type of R-automaton

without auxiliary symbols is shown to be as expressive as the corresponding type of RR-automaton.

This paper is structured as follows. In Section 2 we introduce CD-systems of restarting automata. In Section 3 we define the new mode of operation and give an example. Then in Section 4 we compare the expressive power of CD-systems of restarting automata working in mode ed to that of CD-systems of the same type working in other modes of operation. In Section 5 we carry our investigations over to the various types of deterministic CD-systems of restarting automata, and in Section 6 we establish the equivalence between locally deterministic CD-R(W)(W)-systems and locally deterministic CD-RR(W)(W)-systems working in mode ed. The paper closes with a number of open problems in Section 7.

2 Definitions

We first describe in short the types of restarting automata we will be dealing with. Then we restate the definition of a CD-system of restarting automata from [5].

A *one-way restarting automaton*, abbreviated as RRWW-*automaton*, is a one-tape machine that is described by an 8-tuple $M = (Q, \Sigma, \Gamma, \mathbb{c}, \$, q_0, k, \delta)$, where Q is a finite set of states, Σ is a finite input alphabet, Γ is a finite tape alphabet containing Σ, the symbols $\mathbb{c}, \$ \notin \Gamma$ serve as markers for the left and right border of the work space, respectively, $q_0 \in Q$ is the initial state, $k \geq 1$ is the size of the *read/write window*, and δ is the *transition relation* that associates a finite set of *transition steps* to each pair (q, u) consisting of a state $q \in Q$ and a possible contents u of the read/write window. There are four types of transition steps: *move-right steps*, *rewrite steps*, *restart steps*, and *accept steps*. However, the behaviour of M can be described more succinctly through a finite set of so-called *meta-instructions* (see below).

A *configuration* of M is described by a string $\alpha q \beta$, where $q \in Q$, and either $\alpha = \varepsilon$ (the empty word) and $\beta \in \{\mathbb{c}\} \cdot \Gamma^* \cdot \{\$\}$ or $\alpha \in \{\mathbb{c}\} \cdot \Gamma^*$ and $\beta \in \Gamma^* \cdot \{\$\}$; here q represents the current state, $\alpha\beta$ is the current content of the tape, and it is understood that the head scans the first k symbols of β or all of β when $|\beta| \leq k$. A *restarting configuration* is of the form $q_0 \mathbb{c} w \$$, where $w \in \Gamma^*$; if $w \in \Sigma^*$, then $q_0 \mathbb{c} w \$$ is an *initial configuration*.

A *rewriting meta-instruction* for M has the form $(E_1, u \to v, E_2)$, where E_1 and E_2 are regular expressions, and $u, v \in \Gamma^*$ are words satisfying $k \geq |u| > |v|$. To execute a cycle M chooses a meta-instruction of the form $(E_1, u \to v, E_2)$. On trying to execute this meta-instruction M will get stuck (and so reject) starting from the *restarting configuration* $q_0 \mathbb{c} w \$$, if w does not admit a factorization of the form $w = w_1 u w_2$ such that $\mathbb{c} w_1 \in L(E_1)$ and $w_2 \$ \in L(E_2)$. On the other hand, if w does have factorizations of this form, then one such factorization is chosen nondeterministically, and $q_0 \mathbb{c} w \$$ is transformed into the restarting configuration $q_0 \mathbb{c} w_1 v w_2 \$$. This computation, which is called a *cycle*, is expressed as $w \vdash_M^c$

w_1vw_2. In order to describe the tails of accepting computations we use *accepting meta-instructions* of the form (E_1, Accept), which simply accepts the strings from the regular language $L(E_1)$.

A computation of M now consists of a finite sequence of cycles that is followed by a tail computation. An input word $w \in \Sigma^*$ is *accepted* by M, if there is a computation of M which starts with the initial configuration $q_0\mathbb{c}w\$$, and which finishes by executing an accepting meta-instruction. By $L(M)$ we denote the language consisting of all words accepted by M.

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

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

Obviously, a rewriting meta-instruction for an RWW-automaton has the form $(E_1, u \to v, \Gamma^* \cdot \$)$, which will be abbreviated as $(E_1, u \to v)$.

A *cooperating distributed system of* RRWW-*automata* (or a CD-RRWW-system for short) consists of a finite collection $\mathcal{M} := ((M_i, \sigma_i)_{i\in I}, I_0)$ of RRWW-automata $M_i = (Q_i, \Sigma, \Gamma_i, \mathbb{c}, \$, q_0^{(i)}, k, \delta_i)$ $(i \in I)$, *successor relations* $\sigma_i \subseteq I$ $(i \in I)$, and a subset $I_0 \subseteq I$ of *initial indices*. Here it is required that $Q_i \cap Q_j = \emptyset$ for all $i, j \in I$, $i \neq j$, that $I_0 \neq \emptyset$, that $\sigma_i \neq \emptyset$ for all $i \in I$, and that $i \notin \sigma_i$ for all $i \in I$. Further, let m be one of the following *modes of operation*, where $j \geq 1$:

$= j$: execute exactly j cycles;
$\leq j$: execute at most j cycles;
$\geq j$: execute at least j cycles;
t : continue until no more cycle can be executed.

The computation of $\mathcal{M}$ in mode $= j$ $(\leq j, \geq j)$ on an input word w proceeds as follows. First an index $i_0 \in I_0$ is chosen nondeterministically. Then the RRWW-automaton M_{i_0} starts the computation with the initial configuration $q_0^{(i_0)}\mathbb{c}w\$$, and executes j (at most j, at least j) cycles. Thereafter an index $i_1 \in \sigma_{i_0}$ is chosen nondeterministically, and M_{i_1} continues the computation by executing (at most, at least) j cycles. This continues until, for some $l \geq 0$, the machine M_{i_l} accepts. Should at some stage the chosen machine M_{i_l} be unable to execute the required number of cycles, then the computation fails.

In mode t the chosen automaton M_{i_l} continues with the computation until it either accepts, in which case $\mathcal{M}$ accepts, or until it can neither execute another

cycle nor an accepting tail, in which case an automaton $M_{i_{l+1}}$ with $i_{l+1} \in \sigma_{i_l}$ takes over. Should this machine not be able to execute a cycle or an accepting tail, then the computation of $\mathcal{M}$ fails.

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

3 A New Mode of Operation

Let $\mathcal{M} := ((M_i, \sigma_i)_{i \in I}, I_0)$ be a CD-RRWW-system, and let S_i, T_i be regular expressions that are associated to the component automaton M_i ($i \in I$). The expression S_i is called the *enable condition* of M_i, while T_i is the *disable condition* (or *termination condition*) of M_i. We say that $\mathcal{M}$ is working in mode ed with regular constraints $(S_i, T_i)_{i \in I}$ if, given an input word $w \in \Sigma^*$, $\mathcal{M}$ proceeds as follows:

- An initial index $i_0 \in I_0$ is chosen nondeterministically. If the current tape content (that is, the string $\mathrm{¢}w\$$) does not belong to the regular language $L(S_{i_0})$, then the computation fails. Otherwise, M_{i_0} applies one of its meta-instructions. If that meta-instruction is accepting, then M_{i_0} accepts, and so does $\mathcal{M}$. Otherwise M_{i_0} executes a cycle of the form $w \vdash^c_{M_{i_0}} w_1$. If $\mathrm{¢}w_1\$$ does not belong to the disable language $L(T_{i_0})$, then M_{i_0} continues with another cycle. This process continues until either M_{i_0} accepts, in which case $\mathcal{M}$ accepts the given input, or until M_{i_0} gets stuck (that is, it cannot apply another meta-instruction anymore), in which case the current computation of $\mathcal{M}$ fails, or until the tape content $\mathrm{¢}w_i\$$ produced belongs to the language $L(T_{i_0})$, in which case the current computation of M_{i_0} terminates.
- In the latter case a successor index $i_1 \in \sigma_{i_0}$ is chosen nondeterministically. If the current tape content (that is, the string $\mathrm{¢}w_i\$$) does not belong to the regular language $L(S_{i_1})$, then the computation fails. Otherwise, the computation continues with M_{i_1} as described above. This continues until either the active component accepts, or until the computation fails.

By $L_{\mathsf{ed}}(\mathcal{M})$ we denote the language consisting of all words $w \in \Sigma^*$ that the CD-system $\mathcal{M}$ accepts in mode ed. Further, by $\mathcal{L}_{\mathsf{ed}}(\text{CD-RRWW})$ we denote the class of languages that are accepted by CD-RRWW-systems working in mode ed.

Example 1. Let $\mathcal{M} := ((M_i, \sigma_i)_{i \in I}, I_0)$ be the CD-RW-system that is specified by $I := \{0, 1, 2\}$, $I_0 := \{0\}$, $\sigma_0 := \{1\}$, $\sigma_1 := \{2\}$, $\sigma_2 := \{0\}$, where the RW-automata M_0, M_1, and M_2 are given through the following meta-instructions. Here $\Sigma_0 := \{a, b\}$ and $x, y \in \Sigma_0$:

$$M_0 : (\mathfrak{c} \cdot ((\Sigma_0^2)^+ \cdot xy \cdot \#)^+ \cdot (\Sigma_0^2)^+, xy \cdot \$ \to x \cdot \$),$$
$$\quad (\mathfrak{c} \cdot (xy \cdot \#)^+ \cdot xy \cdot \$, \mathsf{Accept}),$$
$$M_1 : (\mathfrak{c} \cdot ((\Sigma_0^2)^+ \cdot \Sigma_0 \cdot \#)^* \cdot (\Sigma_0^2)^+, xy \cdot \# \to x \cdot \#),$$
$$M_2 : (\mathfrak{c} \cdot ((\Sigma_0^2)^+ \cdot \#)^* \cdot (\Sigma_0^2)^+, x \cdot \# \to \#),$$
$$\quad (\mathfrak{c} \cdot ((\Sigma_0^2)^+ \cdot \#)^+ \cdot (\Sigma_0^2)^+, x \cdot \$ \to \$).$$

In [7] Proposition 13 it is shown that $L_{\mathsf{t}}(\mathcal{M})$ coincides with the *iterated copy language* $L_{\text{copy}^*} := \{\, w(\#w)^n \mid w \in (\Sigma_0^2)^+, n \geq 1 \,\}$. Now we assign regular enable and disable conditions to the components of the system $\mathcal{M}$:

$$S_0 := \mathfrak{c} \cdot ((\Sigma_0^2)^+ \cdot \#)^+ \cdot (\Sigma_0^2)^+ \cdot \$,$$
$$T_0 := \mathfrak{c} \cdot ((\Sigma_0^2)^+ \cdot \#)^+ \cdot (\Sigma_0^2)^+ \cdot \Sigma_0 \cdot \$,$$
$$S_1 := T_0,$$
$$T_1 := \mathfrak{c} \cdot ((\Sigma_0^2)^+ \cdot \Sigma_0 \cdot \#)^+ \cdot (\Sigma_0^2)^+ \cdot \Sigma_0 \cdot \$,$$
$$S_2 := T_1,$$
$$T_2 := S_0.$$

Observe that $L_{\text{copy}^*} \subseteq L(S_0)$ holds. Given an input word $w \in \Sigma^*$, $\mathcal{M}$ will reject immediately if $\mathfrak{c} \cdot w \cdot \$ \notin L(S_0)$; otherwise, w can be written as $w = w_0 \# w_1 \# \dots \# w_n$ for some words $w_0, \dots, w_n \in (\Sigma_0^2)^+$ and some integer $n \geq 1$. If all factors w_i are of length 2, and if they all coincide, then $x \in L_{\text{copy}^*}$, and M_0 accepts in a tail computation. If all factors are of length at least 4, and if they all end with the same suffix xy of length 2, then M_0 executes the cycle

$$w = w_0 \# w_1 \# \dots \# w_n \vdash_{M_0}^c w_0 \# w_1 \# \dots \# z_n x =: x_1,$$

where $w_i = z_i xy$ for all $i = 0, \dots, n$. In all remaining cases M_0 will reject, but then $w \notin L_{\text{copy}^*}$ anyway.

So let us assume that M_0 executes the cycle above. As $\mathfrak{c} \cdot x_1 \cdot \$ \in L(T_0)$, M_0 terminates after this cycle, and M_1 takes over. As $S_1 = T_0$, the enable condition of M_1 is satisfied, and M_1 performs the following computation:

$$x_1 = w_0 \# w_1 \# \dots \# z_n x \vdash_{M_1}^{c^*} z_0 x \# z_1 x \# \dots \# z_n x =: x_2.$$

As $\mathfrak{c} \cdot x_2 \cdot \$ \in L(T_1)$, M_1 now terminates, and M_2 takes over, which is possible as $S_2 = T_1$ holds. Now M_2 performs the computation

$$x_2 = z_0 x \# z_1 x \# \dots \# z_n x \vdash_{M_2}^{c^*} z_0 \# z_1 \# \dots \# z_n =: x_3,$$

and as $\mathfrak{c} \cdot x_3 \cdot \$ \in L(T_2)$, M_2 then terminates. Now M_0 takes over again, and as $T_2 = S_0$, this is indeed possible. Inductively it follows that the input word w is accepted if and only if w belongs to the language L_{copy^*}, that is, it follows that $L_{\mathsf{ed}}(\mathcal{M}) = L_{\text{copy}^*}$ holds. Thus, using the given enable and disable conditions $\mathcal{M}$ accepts in mode ed the same language that it accepts in mode t.

Below we will see that this is not a coincidence, but that in fact mode ed computations can always simulate the mode t computations of a CD-system of restarting automata.

4 On the Power of Enable and Disable Conditions

First we compare the expressive power of mode ed computations to that of mode t computations.

Theorem 1. *Let* $\mathcal{M} := ((M_i, \sigma_i)_{i\in I}, I_0)$ *be a* CD-X*-system for any* $\mathsf{X} \in \{\mathsf{R}, \mathsf{RR}, \mathsf{RW}, \mathsf{RRW}, \mathsf{RWW}, \mathsf{RRWW}\}$. *Then there exists a collection of regular enable and disable conditions* $(S_i, T_i)_{i\in I}$ *such that, with respect to these conditions, the languages* $L_{\mathsf{ed}}(\mathcal{M})$ *and* $L_{\mathsf{t}}(\mathcal{M})$ *coincide.*

Proof. Let $\mathcal{M} = ((M_i, \sigma_i)_{i\in I}, I_0)$ be a CD-X-system, where each component M_i is given through a sequence of meta-instructions $(I_0^{(i)}, I_1^{(i)}, \ldots, I_{m_i}^{(i)})$. Here we can assume without loss of generality that $I_0^{(i)} = (E_0^{(i)}, \mathsf{Accept})$ is the only accepting meta-instruction of M_i, and that the rewriting meta-instruction $I_j^{(i)}$ is of the form $(E_j^{(i)}, u_j^{(i)} \to v_j^{(i)}, F_j^{(i)})$ for all $1 \leq j \leq m_i$, and all $i \in I$. We now define the enable and disable conditions as follows:

$$S_i := E_0^{(i)} \cup \bigcup_{\mu=1}^{m_i} (E_\mu^{(i)} \cdot u_\mu^{(i)} \cdot F_\mu^{(i)}),$$
$$T_i := (¢ \cdot \Gamma^* \cdot \$) \cap ((\Gamma \cup \{¢, \$\})^* \setminus S_i).$$

Then, for $x \in \Gamma^*$, we see that $¢ \cdot x \cdot \$$ belongs to the set $L(S_i)$ if and only there exists a meta-instruction of M_i that is applicable to the restarting configuration $q_0^{(i)}¢x\$$. Thus, if component M_i is called, then the enable condition S_i just ensures that M_i can apply a meta-instruction to the current restarting configuration. Further, the string $¢x'\$$ belongs to the set $L(T_i)$ if and only if no meta-instruction of M_i applies to the restarting copnfiguration $q_0^{(i)}¢x'\$$. Thus, this disable condition is met at the end of a cycle of M_i if and only if M_i is not able to execute another meta-instruction. But this shows that under these enable and disable conditions the mode ed computations of $\mathcal{M}$ coincide with the mode t computations of $\mathcal{M}$. Thus, $L_{\mathsf{ed}}(\mathcal{M}) = L_{\mathsf{t}}(\mathcal{M})$ follows. □

An analogous result can be established for the $= 1$ mode.

Theorem 2. *Let* $\mathcal{M} := ((M_i, \sigma_i)_{i\in I}, I_0)$ *be a* CD-X*-system for any* $\mathsf{X} \in \{\mathsf{R}, \mathsf{RR}, \mathsf{RW}, \mathsf{RRW}, \mathsf{RWW}, \mathsf{RRWW}\}$. *Then there exists a collection of regular enable and disable conditions* $(S_i, T_i)_{i\in I}$ *such that, with respect to these conditions, the languages* $L_{\mathsf{ed}}(\mathcal{M})$ *and* $L_{=1}(\mathcal{M})$ *coincide.*

Proof. Let $\mathcal{M} = ((M_i, \sigma_i)_{i\in I}, I_0)$ be a CD-X-system, where each component M_i is given through a sequence of meta-instructions $(I_0^{(i)}, I_1^{(i)}, \ldots, I_{m_i}^{(i)})$. As above we can assume that $I_0^{(i)} = (E_0^{(i)}, \mathsf{Accept})$ is the only accepting meta-instruction of M_i, and that the rewriting meta-instruction $I_j^{(i)}$ is of the form $(E_j^{(i)}, u_j^{(i)} \to v_j^{(i)}, F_j^{(i)})$ for all $1 \leq j \leq m_i$, and all $i \in I$. We now define the enable and disable conditions as follows:

$$S_i := E_0^{(i)} \cup \bigcup_{\mu=1}^{m_i} (E_\mu^{(i)} \cdot u_\mu^{(i)} \cdot F_\mu^{(i)}),$$
$$T_i := (¢ \cdot \Gamma^* \cdot \$).$$

Then, for $x \in \Gamma^*$, we see that $\mathbb{c} \cdot x \cdot \$$ belongs to the set $L(S_i)$ if and only there exists a meta-instruction of M_i that is applicable to the restarting configuration $q_0^{(i)}\mathbb{c}x\$$. Thus, if component M_i is called, then the enable condition S_i just ensures that M_i can apply a meta-instruction to the current restarting configuration. Further, each string $\mathbb{c}y\$$ with $y \in \Gamma^*$ belongs to the set $L(T_i)$. Thus, this disable condition is met at the end of the first cycle of M_i. This shows that under these enable and disable conditions the mode ed computations of $\mathcal{M}$ coincide with the mode $= 1$ computations of $\mathcal{M}$. Thus, $L_{\mathsf{ed}}(\mathcal{M}) = L_{=1}(\mathcal{M})$ follows. □

A CD-system of restarting automata that is working in mode $= j$ can also be simulated by a CD-system of restarting automata that is working in mode ed. However, the simulating system has several component automata for each component of the system being simulated.

Theorem 3. *Let $\mathcal{M} := ((M_i, \sigma_i)_{i\in I}, I_0)$ be a $\mathsf{CD\text{-}X}$-system for any $\mathsf{X} \in \{\mathsf{R}, \mathsf{RR}, \mathsf{RW}, \mathsf{RRW}, \mathsf{RWW}, \mathsf{RRWW}\}$, and let $j \geq 2$. Then there exists a $\mathsf{CD\text{-}X}$-system $\mathcal{M}' := ((M'_{(i,\mu)}, \sigma'_{(i,\mu)})_{(i,\mu)\in I\times\{1,\ldots,j\}}, I'_0)$ and regular enable and disable conditions $(S_{(i,\mu)}, T_{(i,\mu)})$, $(i,\mu) \in I \times \{1,\ldots,j\}$, such that the languages $L_{\mathsf{ed}}(\mathcal{M}')$ and $L_{=j}(\mathcal{M})$ coincide.*

Proof. Let $\mathcal{M} := ((M_i, \sigma_i)_{i\in I}, I_0)$ be a $\mathsf{CD\text{-}X}$-system, and let $j \geq 2$. Again we assume that each component M_i is given through a sequence of meta-instructions $(I_0^{(i)}, I_1^{(i)}, \ldots, I_{m_i}^{(i)})$, where $I_0^{(i)} = (E_0^{(i)}, \mathsf{Accept})$ is the only accepting meta-instruction of M_i, and the rewriting meta-instruction $I_\nu^{(i)}$ is of the form $(E_\nu^{(i)}, u_\nu^{(i)} \to v_\nu^{(i)}, F_\nu^{(i)})$ for all $1 \leq \nu \leq m_i$, and all $i \in I$. For all $i \in I$, we define the automata $M'_{(i,\mu)}$ $(1 \leq \mu \leq j)$ as j disjoint copies of the component M_i of $\mathcal{M}$. Further, we take $I'_0 := \{\, (i,1) \mid i \in I_0 \,\}$ as set of initial indices, and we define the successor sets $\sigma'_{(i,\mu)}$ as follows:

$$\begin{aligned} \sigma'_{(i,\mu)} &:= \{(i, \mu+1)\} && \text{for all } i \in I \text{ and } \mu = 1,\ldots,j-1,\\ \sigma'_{(i,j)} &:= \{\, (l,1) \mid l \in \sigma_i \,\} && \text{for all } i \in I. \end{aligned}$$

Finally, we introduce the following enable and disable conditions:

$$\begin{aligned} S_{(i,\mu)} &:= E_0^{(i)} \cup \textstyle\bigcup_{\nu=1}^{m_i}(E_\nu^{(i)} \cdot u_\nu^{(i)} \cdot F_\nu^{(i)}) && \text{for all } i \in I \text{ and all } \mu = 1,\ldots,j,\\ T_{(i,\mu)} &:= (\mathbb{c} \cdot \Gamma^* \cdot \$) && \text{for all } i \in I \text{ and all } \mu = 1,\ldots,j. \end{aligned}$$

The enable conditions just ensure that $M_{(i,\mu)}$ can apply a meta-instruction to the current restarting configuration. Further, the disable condition is always met at the end of the first cycle of $M_{(i,\mu)}$. Together with the revised successor relations this shows that under these enable and disable conditions the mode ed computations of $\mathcal{M}'$ simulate exactly the mode $= j$ computations of $\mathcal{M}$. Thus, we see that $L_{\mathsf{ed}}(\mathcal{M}') = L_{=j}(\mathcal{M})$ holds. □

To simulate the $\leq j$ mode by the ed mode we simply take the construction from the proof of the previous theorem, but we change the successor relations

as follows:

$$\begin{aligned}
\sigma'_{(i,\mu)} &:= \{\,(l,1) \mid l \in \sigma_i\,\} \cup \{(i,\mu+1)\} && \text{for all } i \in I \text{ and } \mu = 1,\ldots,j-1,\\
\sigma'_{(i,j)} &:= \{\,(l,1) \mid l \in \sigma_i\,\} && \text{for all } i \in I.
\end{aligned}$$

This gives the following result.

Corollary 1. *Let $\mathcal{M} := ((M_i,\sigma_i)_{i\in I}, I_0)$ be a* CD-X*-system for any* $\mathsf{X} \in \{\mathsf{R},\mathsf{RR},\mathsf{RW},\mathsf{RRW},\mathsf{RWW},\mathsf{RRWW}\}$*, and let $j \geq 2$. Then there exists a* CD-X*-system $\mathcal{M}' := ((M'_{(i,\mu)},\sigma'_{(i,\mu)})_{(i,\mu)\in I\times\{1,\ldots,j\}}, I'_0)$ and regular enable and disable conditions $(S_{(i,\mu)},T_{(i,\mu)})$, $(i,\mu) \in I\times\{1,\ldots,j\}$, such that the languages $L_{\mathsf{ed}}(\mathcal{M}')$ and $L_{\leq j}(\mathcal{M})$ coincide.*

Finally, to simulate the $\geq j$ mode by the ed mode we replace each component M_i of the system $\mathcal{M}$ by $j+1$ components, and define the successor relations as follows:

$$\begin{aligned}
\sigma'_{(i,\mu)} &:= \{(i,\mu+1)\} && \text{for all } i \in I \text{ and } \mu = 1,\ldots,j-1,\\
\sigma'_{(i,j)} &:= \{\,(l,1) \mid l \in \sigma_i\,\} \cup \{(i,j+1)\} && \text{for all } i \in I,\\
\sigma'_{(i,j+1)} &:= \{\,(l,1) \mid l \in \sigma_i\,\} \cup \{(i,j)\} && \text{for all } i \in I.
\end{aligned}$$

Corollary 2. *Let $\mathcal{M} := ((M_i,\sigma_i)_{i\in I}, I_0)$ be a* CD-X*-system for any* $\mathsf{X} \in \{\mathsf{R},\mathsf{RR},\mathsf{RW},\mathsf{RRW},\mathsf{RWW},\mathsf{RRWW}\}$*, and let $j \geq 1$. Then there exists a* CD-X*-system $\mathcal{M}' := ((M'_{(i,\mu)},\sigma'_{(i,\mu)})_{(i,\mu)\in I\times\{1,\ldots,j+1\}}, I'_0)$ and regular enable and disable conditions $(S_{(i,\mu)},T_{(i,\mu)})$, $(i,\mu) \in I\times\{1,\ldots,j+1\}$, such that the languages $L_{\mathsf{ed}}(\mathcal{M}')$ and $L_{\geq j}(\mathcal{M})$ coincide.*

Thus, the ed mode is sufficiently powerful to simulate all the other modes considered so far. On the other hand, it can be shown that for CD-RR(W)(W)-systems, the $=1$ mode is as expressive as the ed mode.

Theorem 4. *Let $\mathcal{M} := ((M_i,\sigma_i)_{i\in I}, I_0)$ be a* CD-X*-system for any* $\mathsf{X} \in \{\mathsf{RR},\mathsf{RRW},\mathsf{RRWW}\}$*, and let $(S_i,T_i,)$, $i \in I$, be a collection of regular enable and disable conditions for $\mathcal{M}$. Then there exists a* CD-X*-system $\mathcal{M}' := ((M'_i,\sigma'_i)_{i\in I'}, I'_0)$ such that the languages $L_{=1}(\mathcal{M}')$ and $L_{\mathsf{ed}}(\mathcal{M})$ coincide.*

Proof. Let $\mathcal{M} := ((M_i,\sigma_i)_{i\in I}, I_0)$ be a CD-X-system, and let $(S_i,T_i,)$, $i \in I$, be a collection of regular enable and disable conditions for $\mathcal{M}$. Let $n := |I|$ be the number of components of $\mathcal{M}$. We can assume without loss of generality that $I = \{1,\ldots,n\}$.

We define a CD-X-system $\mathcal{M}' := ((M'_i,\sigma'_i)_{i\in I'}, I'_0)$ as follows. Each component M_i of $\mathcal{M}$ will be simulated by $n+2$ components of $\mathcal{M}'$. In addition, we need three components for each initial component of $\mathcal{M}$. Accordingly, we take

$$I' := I \times \{1,\ldots,n+2\} \cup \{\,(\hat{i},j) \mid i \in I_0,\ 1 \leq j \leq 3\,\}.$$

For each $i \in I$ and each $j \in \{1,\ldots,n+2\}$, $M'_{(i,j)}$ is a copy of M_i, and for each $i \in I_0$ and each $j \in \{1,2,3\}$, $M'_{(\hat{i},j)}$ is another copy of M_i. These components are now modified as follows:

- For $j = 1, \ldots, n$, component $M'_{(i,j)}$ verifies with each of its meta-instructions that the tape content $¢w\$$ of the actual restarting configuration belongs to the regular set $L(S_i) \cap L(T_j)$, that is, that it satisfies the enable condition of component M_i and the disable condition of component M_j. In the affirmative, it just executes the current meta-instruction of M_i, but in the negative it halts and rejects.
- For $j = n+1, n+2$, component $M'_{(i,j)}$ verifies with each of its meta-instructions that the tape content $¢w\$$ of the actual restarting configuration does not belong to the regular set $L(T_i)$, that is, that it does not satisfy the disable condition of component M_i. In the affirmative, it just executes the current meta-instruction of M_i, but in the negative it halts and rejects.
- For $i \in I_0$, component $M'_{(\hat{i},1)}$ verifies with each of its meta-instructions that the tape content $¢w\$$ of the actual restarting configuration belongs to the regular set $L(S_i)$, that is, that it satisfies the enable condition of component M_i. Components $M'_{(\hat{i},2)}$ and $M'_{(\hat{i},3)}$ verify with each of their meta-instructions that the tape content $¢w\$$ of the actual restarting configuration does not belong to the regular set $L(T_i)$, that is, that it does not satisfy the disable condition of component M_i.

For the initial indices of $\mathcal{M}'$ be take the set $I'_0 := \{ (\hat{i}, 1) \mid i \in I_0 \}$, and we define the successor relations as follows:

$$
\begin{aligned}
\sigma'_{(i,j)} &:= \{(i, n+1)\} \cup \{ (l, i) \mid l \in \sigma_i \} && \text{for all } i \in I \text{ and all } 1 \leq j \leq n,\\
\sigma'_{(i,n+1)} &:= \{(i, n+2)\} \cup \{ (l, i) \mid l \in \sigma_i \} && \text{for all } i \in I,\\
\sigma'_{(i,n+2)} &:= \{(i, n+1)\} \cup \{ (l, i) \mid l \in \sigma_i \} && \text{for all } i \in I,\\
\sigma'_{(\hat{i},1)} &:= \{(\hat{i}, 2)\} \cup \{ (l, i) \mid l \in \sigma_i \} && \text{for all } i \in I_0,\\
\sigma'_{(\hat{i},2)} &:= \{(\hat{i}, 3)\} \cup \{ (l, i) \mid l \in \sigma_i \} && \text{for all } i \in I_0,\\
\sigma'_{(\hat{i},3)} &:= \{(\hat{i}, 2)\} \cup \{ (l, i) \mid l \in \sigma_i \} && \text{for all } i \in I_0.
\end{aligned}
$$

Now, given an input $w \in \Sigma^*$, an index $i_0 \in I_0$ is chosen, and component M_{i_0} begins the computation of $\mathcal{M}$ on input w by executing a certain number of cycles. Actually, it is first checked whether $¢w\$$ belongs to the regular set $L(S_{i_0})$, and then M_{i_0} continues with the computation until it either terminates (accepting or non-accepting) or until the tape content $¢w_1\$$ obtained belongs to the regular set $L(T_{i_0})$. Then an index $i_1 \in \sigma_{i_0}$ is chosen, and component M_{i_1} continues with the computation, provided that the tape content belongs to the regular set $L(S_{i_1})$. This continues until the actual component terminates.

Now let us consider the possible mode $= 1$ computations of $\mathcal{M}'$ on input w. Here we can choose the initial component $M'_{(\hat{i}_0,1)}$. It verifies that the tape content belongs to the regular set $L(S_{i_0})$, and in the affirmative it executes the same cycle as M_{i_0}. Then $M'_{(\hat{i}_0,2)}$ becomes active. It checks that the current tape content does not belong to the regular set $L(T_{i_0})$, and in the affirmative it executes the same cycle as M_{i_0}. Thereafter $M'_{(\hat{i}_0,3)}$ becomes active. Thus, by alternating between the latter two components the computation of M_{i_0} described above is being

simulated. At some stage the index $(i_1, i_0) \in \sigma'_{(\hat{i}_0,j)}$ $(j \in \{1,2,3\})$ is chosen, and component $M'_{(i_1,i_0)}$ continues with the computation. It verifies that the actual tape content belongs to the regular set $L(S_{i_1})$ as well as to the regular set $L(T_{i_0})$, which corresponds to the situation in the above computation of $\mathcal{M}$ when component M_{i_1} takes over from component M_{i_0}. If these constraints are met, then $M'_{(i_1,i_0)}$ executes the same cycle as M_{i_1}, and then $M'_{(i_1,n+1)}$ becomes active. It checks that the current tape content does not belong to the regular set $L(T_{i_1})$, and in the affirmative it executes the next cycle of M_{i_1}. It follows that the computations of $\mathcal{M}'$ in mode $= 1$ can simulate all the mode ed computations of $\mathcal{M}$.

Conversely, it can be shown that the mode $= 1$ computations of $\mathcal{M}'$ can only simulate mode ed computations of $\mathcal{M}$. It follows that $L_{=1}(\mathcal{M}') = L_{\mathsf{ed}}(\mathcal{M})$ holds. □

Thus, we have the following consequence.

Corollary 3. *For all* $\mathsf{X} \in \{\mathsf{RR}, \mathsf{RRW}, \mathsf{RRWW}\}$, $\mathcal{L}_{\mathsf{ed}}(\mathsf{CD\text{-}X}) = \mathcal{L}_{=1}(\mathsf{CD\text{-}X})$.

It remains open whether Theorem 4 extends to $\mathsf{CD\text{-}R(W)(W)}$-systems. Thus, it is not known yet whether the inclusion $\mathcal{L}_{=1}(\mathsf{CD\text{-}R(W)(W)}) \subseteq \mathcal{L}_{\mathsf{ed}}(\mathsf{CD\text{-}R(W)(W)})$ of Theorem 2 is proper or not.

5 Deterministic CD-Systems of Restarting Automata

Various forms of determinism have been studied for CD-systems of restarting automata [7]. Here we restate the relevant definitions in short.

A CD-system $\mathcal{M} := ((M_i, \sigma_i)_{i \in I}, I_0)$ of restarting automata is called *locally deterministic* if M_i is a deterministic restarting automaton for each $i \in I$. As the successor system is chosen nondeterministically from among all systems M_j with $j \in \sigma_i$, computations of a locally deterministic CD-system of restarting automata are in general not completely deterministic.

To avoid this remaining nondeterminism the following variant of determinism has been introduced. A CD-system $\mathcal{M} := ((M_i, \sigma_i)_{i \in I}, I_0)$ is called *strictly deterministic* if I_0 is a singleton, if M_i is a deterministic restarting automaton, if $|\sigma_i| = 1$ for each $i \in I$, and if the function $\sigma : I \to I$ that maps each component to its unique successor is a bijection.

However, the restriction of having at most a single possible successor for each component system is a rather serious one. Thus, a third notion has been defined. A CD-system $\mathcal{M} := ((M_i, \sigma_i)_{i \in I}, I_0)$ is called *globally deterministic* if I_0 is a singleton, if M_i is a deterministic restarting automaton for each $i \in I$, and if, for each $i \in I$, each restart operation of M_i is combined with an index from the set σ_i. Thus, when M_i finishes a part of a computation according to the actual mode of operation by executing the restart operation $\delta_i(q, u) = (\mathsf{Restart}, j)$, where $j \in \sigma_i$, then the component M_j takes over. In this way it is guaranteed that all computations of a globally deterministic CD-system are deterministic. However, for a component system M_i there can still be several possible successor systems.

This is reminiscent of the way in which nonforgetting restarting automata (see, e.g., [8]) work.

We use the prefix det-local- to denote locally deterministic CD-systems, the prefix det-global- to denote globally deterministic CD-systems, and the prefix det-strict- to denote strictly deterministic CD-systems. For each type of restarting automaton $\mathsf{X} \in \{\mathsf{R}, \mathsf{RR}, \mathsf{RW}, \mathsf{RRW}, \mathsf{RWW}, \mathsf{RRWW}\}$, the following inclusions hold [7]:

$$\begin{aligned}\mathcal{L}(\text{det-X}) \subseteq \mathcal{L}_{=1}(\text{det-strict-CD-X}) \subseteq \mathcal{L}_{=1}(\text{det-global-CD-X})\\ \subseteq \mathcal{L}_{=1}(\text{det-local-CD-X}) \subseteq \mathcal{L}_{=1}(\text{CD-X}).\end{aligned}$$

Concerning the power of enable and disable conditions for the various types of deterministic CD-systems we have the following results.

Corollary 4. *Let $\mathcal{M} := ((M_i, \sigma_i)_{i\in I}, I_0)$ be a* CD-X*-system that is strictly (globally, locally) deterministic, where $\mathsf{X} \in \{\mathsf{R}, \mathsf{RR}, \mathsf{RW}, \mathsf{RRW}, \mathsf{RWW}, \mathsf{RRWW}\}$. Then there exists a collection of regular enable and disable conditions $(S_i, T_i)_{i\in I}$ such that, with respect to these conditions, the languages $L_{\mathsf{ed}}(\mathcal{M})$ and $L_{\mathsf{t}}(\mathcal{M})$ coincide.*

Proof. In the proof of Theorem 1 neither the component automata nor the successor relations are modified. Thus, it carries over to all types of deterministic CD-X-systems. □

Analogously the following result is obtained.

Corollary 5. *Let $\mathcal{M} := ((M_i, \sigma_i)_{i\in I}, I_0)$ be a* CD-X*-system that is strictly (globally, locally) deterministic, where $\mathsf{X} \in \{\mathsf{R}, \mathsf{RR}, \mathsf{RW}, \mathsf{RRW}, \mathsf{RWW}, \mathsf{RRWW}\}$. Then there exists a collection of regular enable and disable conditions $(S_i, T_i)_{i\in I}$ such that, with respect to these conditions, the languages $L_{\mathsf{ed}}(\mathcal{M})$ and $L_{=1}(\mathcal{M})$ coincide.*

For locally deterministic CD-systems we also have the following result.

Corollary 6. *Let $\mathcal{M} := ((M_i, \sigma_i)_{i\in I}, I_0)$ be a* CD-X*-system that is locally deterministic, where $\mathsf{X} \in \{\mathsf{R}, \mathsf{RR}, \mathsf{RW}, \mathsf{RRW}, \mathsf{RWW}, \mathsf{RRWW}\}$, let $j \geq 2$, and let* m *be one of the modes of operation $\leq j$, $= j$, or $\geq j$. Then there exists a locally deterministic* CD-X*-system $\mathcal{M}'$ and regular enable and disable conditions such that the languages $L_{\mathsf{ed}}(\mathcal{M}')$ and $L_{\mathrm{m}}(\mathcal{M})$ coincide.*

Proof. Let $\mathcal{M} := ((M_i, \sigma_i)_{i\in I}, I_0)$ be a locally deterministic CD-X-system, let $j \geq 2$, and let m be one of the modes of operation $\leq j$, $= j$, or $\geq j$. In the proof of Theorem 3 the CD-X-system $\mathcal{M}'$ is constructed from $\mathcal{M}$ by taking a finite number of disjoint copies of all the component automata of $\mathcal{M}$, and by modifying the successor relations and the set of initial indices. Thus, if $\mathcal{M}$ is locally deterministic, then so is $\mathcal{M}'$. The same observation applies to the proofs of Corollary 1 and Corollary 2. □

It remains to consider globally deterministic and strictly deterministic CD-systems. Here we only need to study the mode $= j$ ($j \geq 2$), as the other modes are inherently nondeterministic.

Corollary 7. *Let $\mathcal{M} := ((M_i, \sigma_i)_{i \in I}, I_0)$ be a* CD-X*-system that is strictly (globally) deterministic, where* $\mathsf{X} \in \{\mathsf{R}, \mathsf{RR}, \mathsf{RW}, \mathsf{RRW}, \mathsf{RWW}, \mathsf{RRWW}\}$*, and let $j \geq 2$. Then there exists a strictly (globally) deterministic* CD-X*-system $\mathcal{M}'$ and regular enable and disable conditions such that the languages $L_{\mathsf{ed}}(\mathcal{M}')$ and $L_{=j}(\mathcal{M})$ coincide.*

Proof. For the case of a strictly deterministic CD-X-system $\mathcal{M}$, we can simply use the proof of Theorem 3. If $\mathcal{M}$ is a globally deterministic CD-X-system, then we can still take the same construction. Here, however, we need to describe which successor index is associated with which restart operation of the various component automata.

Consider a component M_i of $\mathcal{M}$. In $\mathcal{M}'$ there are j disjoint copies $M'_{(i,1)}, \ldots, M'_{(i,j)}$ of M_i. For $\mu = 1, \ldots, j-1$, the corresponding successor set $\sigma'_{(i,\mu)}$ is the singleton $\{(i, \mu + 1)\}$, and hence, this index is associated to all restart operations of $M'_{(i,\mu)}$. Finally, $\sigma'_{(i,j)} = \{\, (l, 1) \mid l \in \sigma_i \,\}$. Now to a restart operation $\delta'_{(i,j)}(q_r^{(i,j)}, u) = \mathsf{Restart}$ we associate the successor $(l, 1)$, provided that $l \in \sigma_i$ is the successor that is associated with the corresponding restart operation $\delta_i(q_r^{(i)}, u) = \mathsf{Restart}$ of M_i. Then $\mathcal{M}'$ is globally deterministic, and as before it follows that $L_{\mathsf{ed}}(\mathcal{M}') = L_{=j}(\mathcal{M})$ holds. □

Finally we want to carry Theorem 4 over to the setting of deterministic CD-systems of restarting automata. In the proof of Theorem 4 a CD-RR(W)(W)-system $\mathcal{M}'$ is constructed from a given CD-RR(W)(W)-system $\mathcal{M}$. If all component automata of $\mathcal{M}$ are deterministic, then so are all component automata of $\mathcal{M}'$. Thus, we have the following result.

Corollary 8. *Let $\mathcal{M} := ((M_i, \sigma_i)_{i \in I}, I_0)$ be a locally deterministic* CD-X*-system, where* $\mathsf{X} \in \{\mathsf{RR}, \mathsf{RRW}, \mathsf{RRWW}\}$*, and let $(S_i, T_{i,})$, $i \in I$, be a collection of regular enable and disable conditions for $\mathcal{M}$. Then there exists a locally deterministic* CD-X*-system $\mathcal{M}' := ((M'_i, \sigma'_i)_{i \in I'}, I'_0)$ such that the languages $L_{=1}(\mathcal{M}')$ and $L_{\mathsf{ed}}(\mathcal{M})$ coincide.*

For globally deterministic CD-RRWW-systems we have the corresponding result. However, for these systems we need a different technique for constructing the system $\mathcal{M}'$.

Theorem 5. *Let $\mathcal{M} := ((M_i, \sigma_i)_{i \in I}, I_0)$ be a globally deterministic* CD-X*-system, where* $\mathsf{X} \in \{\mathsf{RR}, \mathsf{RRW}, \mathsf{RRWW}\}$*, and let $(S_i, T_{i,})$, $i \in I$, be a collection of regular enable and disable conditions for $\mathcal{M}$. Then there exists a globally deterministic* CD-X*-system $\mathcal{M}' := ((M'_i, \sigma'_i)_{i \in I'}, I'_0)$ such that the languages $L_{=1}(\mathcal{M}')$ and $L_{\mathsf{ed}}(\mathcal{M})$ coincide.*

Proof. Let $\mathcal{M} := ((M_i, \sigma_i)_{i \in I}, i_0)$ be a globally deterministic CD-X-system, and let $(S_i, T_{i,})$, $i \in I$, be a collection of regular enable and disable conditions for $\mathcal{M}$. Let $n := |I|$ be the number of components of $\mathcal{M}$. We can assume without loss of generality that $I = \{1, \ldots, n\}$.

We define a globally deterministic CD-X-system $\mathcal{M}' := ((M'_i, \sigma'_i)_{i \in I'}, i'_0)$ as follows. Each component M_i of $\mathcal{M}$ will be simulated by three components of $\mathcal{M}'$. Accordingly, we take $I' := I \times \{1, 2, 3\}$, and take $i'_0 := (i_0, 1)$. For each $i \in I$, the three components $M'_{(i,\mu)}$, $1 \leq \mu \leq 3$, are obtained as copies of M_i that are slightly modified as follows:

- $M'_{(i,1)}$ verifies with each of its meta-instructions that the tape content $\mathrm{¢}w\$$ of the actual restarting configuration belongs to regular set $L(S_i)$, that is, that it satisfies the enable condition of component M_i. In addition, it checks with each of its rewriting meta-instructions whether the tape content $\mathrm{¢}w_1vw_2\$$ produced by applying this meta-instruction belongs to the regular set $L(T_i)$, that is, whether it satisfies the disable condition of component M_i. In the affirmative, the restart operation of this meta-instruction is associated to the index $(l, 1)$, where $l \in \sigma_i$ is the index that is associated with the corresponding restart operation of M_i. In the negative, that is, if the resulting tape content does not yet meet the disable condition of M_i, then the restart operation of this meta-instruction is associated to the index $(i, 2)$.
- $M'_{(i,2)}$ checks with each of its rewriting meta-instructions whether the tape content $\mathrm{¢}w_1vw_2\$$ produced by applying this meta-instruction belongs to the regular set $L(T_i)$, that is, whether it satisfies the disable condition of component M_i. In the affirmative, the restart operation of this meta-instruction is associated to the index $(l, 1)$, where $l \in \sigma_i$ is the index that is associated with the corresponding restart operation of M_i. In the negative, that is, if the resulting tape content does not yet meet the disable condition of M_i, then the restart operation of this meta-instruction is associated to the index $(i, 3)$.
- $M'_{(i,3)}$ checks with each of its rewriting meta-instructions whether the tape content $\mathrm{¢}w_1vw_2\$$ produced by applying this meta-instruction belongs to the regular set $L(T_i)$, that is, whether it satisfies the disable condition of component M_i. In the affirmative, the restart operation of this meta-instruction is associated to the index $(l, 1)$, where $l \in \sigma_i$ is the index that is associated with the corresponding restart operation of M_i. In the negative, that is, if the resulting tape content does not yet meet the disable condition of M_i, then the restart operation of this meta-instruction is associated to the index $(i, 2)$.

Thus, $\mathcal{M}'$ is indeed a globally deterministic CD-X-system. Further, it is easily seen that in mode $= 1$ it simulates the mode ed computations of $\mathcal{M}$. It follows that $L_{=1}(\mathcal{M}') = L_{\mathsf{ed}}(\mathcal{M})$ holds. □

It is open whether Corollary 8 and Theorem 5 extend to CD-R(W)(W)-systems. Further, it remains currently open whether a result corresponding to Theorem 5 also holds for strictly deterministic CD-RRWW-systems. At least for strictly deterministic CD-systems of restarting automata without auxiliary symbols, that is, for strictly deterministic CD-RR- and CD-RRW-systems, it does not hold as shown by the following results. Recall the iterated copy language $L_{\mathrm{copy}^*} := \{ w(\#w)^n \mid w \in (\Sigma_0^2)^+, n \geq 1 \}$ from Example 1, where $\Sigma_0 := \{a, b\}$. In [6, 7] the following results are established.

Proposition 1. *L_{copy^*} is accepted by*

- *a globally deterministic* CD-R*-system working in mode* $= 1$,
- *a strictly deterministic* CD-RWW*-system working in mode* $= 1$, *and*
- *a strictly deterministic* CD-R*-system working in mode* t,

but it is not accepted by any strictly deterministic CD-RRW*-system working in mode* $= 1$.

Thus, Corollary 4 implies that the language L_{copy^*} is accepted by a strictly deterministic CD-R-system working in mode ed. Together with Corollary 5 this yields the following proper inclusion.

Corollary 9. *For all* $\mathsf{X} \in \{\mathsf{R}, \mathsf{RR}, \mathsf{RW}, \mathsf{RRW}\}$,

$$\mathcal{L}_{=1}(\text{det-strict-CD-X}) \subsetneq \mathcal{L}_{\mathsf{ed}}(\text{det-strict-CD-X}).$$

6 CD-R-Systems Versus CD-RR-Systems

Here we compare the expressive power of CD-RR(W)(W)-systems working in mode ed to that of CD-R(W)(W)-systems working in mode ed. To this end we first study the information that a description by meta-instructions reveals on a deterministic RRWW-automaton.

Let $M = (Q, \Sigma, \Gamma, ¢, \$, q_0, k, \delta)$ be a deterministic RRWW-automaton, and let $I_0 = (E_0, \mathsf{Accept})$ and $I_i = (E_i, u_i \to v_i, E_i')$ $(1 \leq i \leq n)$ be a sequence of meta-instructions that describe the behaviour of M. Here we can assume without loss of generality that $|u_i| = k$ holds for all $i = 1, \ldots, n$. As M is deterministic, the above meta-instructions are used as follows. Assume that M is in the restarting configuration $q_0¢w\$$. Then M scans the tape from left to right until it detects the shortest prefix w_1 of w such that $w_1 = w_3u_i$ and $¢w_3 \in L(E_i)$ for some $i \in \{1, \ldots, n\}$. It then rewrites u_i into v_i and checks whether the corresponding suffix w_2 of w satisfies the condition that $w_2\$ \in L(E_i')$. At the same time it checks whether the original tape content $¢w\$$ belongs to the language $L(E_0)$. If the latter holds, then M halts and accepts; if $¢w\$ \notin L(E_0)$, but $w_2\$ \in L(E_i')$, then M restarts, which yields the restarting configuration $q_0¢w_3v_iw_2\$$. Finally, if $w_2\$ \in L(E_i')$ does not hold, either, then M halts and rejects. If no prefix of the above form is found, then M halts and rejects as well, unless $¢w\$ \in L(E_0)$ holds, in which case M halts and accepts.

Thus, we can replace each regular constraint E_i by a regular constraint F_i such that

$$L(F_i) = \{\, ¢w \in L(E_i) \mid \text{ No proper prefix of } ¢wu_i \text{ is in } \bigcup_{r=1}^{n} (L(E_r) \cdot u_r) \,\},$$

and the resulting meta-instructions $I_i' = (F_i, u_i \to v_i, E_i')$ $(1 \leq i \leq n)$ will describe M together with I_0.

The new contraints have the following advantage. If

$$¢w\$ \in \bigcup_{i=1}^{n} (L(F_i) \cdot u_i \cdot L(E'_i)),$$

then there exist a unique index $i \in \{1, \ldots, n\}$ and a unique factorization $w = w_1 u_i w_2$ such that $¢w_1 \in L(F_i)$ and $w_2\$ \in L(E'_i)$ hold. Thus, if it is known that $¢w\$ \in \bigcup_{i=1}^{n} (L(F_i) \cdot u_i \cdot L(E'_i))$ holds, then on detecting a prefix w_1 of w satisfying $¢w_1u_i \in L(F_i) \cdot u_i$, it is guaranteed that the corresponding suffix w_2 of w satisfies the condition $w_2\$ \in L(E'_i)$. Observe, however, that in general the intersection $L(E_0) \cap \bigcup_{i=1}^{n} (L(F_i) \cdot u_i \cdot L(E'_i))$ will not be empty, that is, some words are accepted by M in tail computations that have a prefix from the language $L(F_i) \cdot u_i$ for some value of i.

We now use the above observation for establishing the following relationship between locally deterministic CD-RR(W)(W)-systems and locally deterministic CD-R(W)(W)-systems working in mode ed.

Theorem 6.
For all $\mathsf{X} \in \{\lambda, \mathsf{W}, \mathsf{WW}\}$, $\mathcal{L}_{\mathsf{ed}}(\mathsf{det\text{-}local\text{-}CD\text{-}RRX}) \subseteq \mathcal{L}_{\mathsf{ed}}(\mathsf{det\text{-}local\text{-}CD\text{-}RX})$.

Proof. Because of Corollary 8 it suffices to consider locally deterministic CD-RRX-systems that are working in mode = 1. Let $\mathcal{M} := ((M_i, \sigma_i)_{i \in I}, I_0)$ be a locally deterministic CD-RRX-system, and let $L = L_{=1}(\mathcal{M})$. Each component M_i can be described by a finite sequence of meta-instructions of the form $(I_{i,0}, I_{i,1}, \ldots, I_{i,n_i})$, where $I_{i,0} = (E_{i,0}, \mathsf{Accept})$ is an accepting meta-instruction, and $I_{i,j} = (E_{i,j}, u_{i,j} \to v_{i,j}, E'_{i,j})$ $(1 \leq j \leq n_i)$ are rewriting meta-instructions.

We now construct a locally deterministic CD-RX-system $\mathcal{M}'$ and enable and disable conditions such that $L_{\mathsf{ed}}(\mathcal{M}') = L$ holds. In this construction we will have two components $P_{(i,a)}$ and $P_{(i,c)}$ for each component M_i. Thus, we take $\mathcal{M}' := ((P_{(i,\mu)}, \sigma_{(i,\mu)})_{i \in I, \mu \in \{a,c\}}, I'_0)$, where

$$\sigma_{(i,a)} := \sigma_{(i,c)} := \{\, (j,a), (j,c) \mid j \in \sigma_i \,\} \text{ for all } i \in I,$$

and

$$I'_0 := \{\, (i,a), (i,c) \mid i \in I_0 \,\}.$$

For each index $i \in I$, the component $P_{(i,a)}$ is described by the accepting meta-instruction $I_{i,0}$. Its enable condition $S_{(i,a)}$ describes the language $¢ \cdot \Gamma^* \cdot \$$, where Γ is the (combined) tape alphabet of the components of $\mathcal{M}$, and its disable condition $T_{(i,a)}$ describes the same language. If this component is called during a computation of $\mathcal{M}'$, then the computation necessarily ends: either the tape content $¢w\$$ belongs to the regular language $L(E_{i,0})$, and $P_{(i,a)}$ accepts, or it does not belong to this language, and then $P_{(i,a)}$ rejects.

It remains to define the components $P_{(i,c)}$ and the enable and disable conditions $S_{(i,c)}$ and $T_{(i,c)}$ for all $i \in I$. Let $i \in I$, and let $j \in \{1, \ldots, n_i\}$. By $F_{i,j}$ we

denote the regular expression for the language

$$L(F_{i,j}) = \{\, ¢w \in L(E_{i,j}) \mid \text{ No proper prefix of } ¢wu_{i,j} \text{ is in } \bigcup_{r=1}^{n_i} (L(E_{i,r}) \cdot u_{i,r}) \,\}.$$

Then we define $P_{(i,c)}$ by the meta-instructions $I'_{i,j} := (F_{i,j}, u_{i,j} \to v_{i,j})$ $(1 \leq j \leq n_i)$, and take $S_{(i,c)}$ and $T_{(i,c)}$ to describe the languages

$$L(S_{(i,c)}) := \bigcup_{j=1}^{n_i} (L(F_{i,j}) \cdot u_{i,j} \cdot L(E'_{i,j})), \text{ and } L(T_{(i,c)}) := ¢ \cdot \Gamma^* \cdot \$.$$

If this component is called during a computation of $\mathcal{M}'$, and if $¢w\$$ is the corresponding restarting configuration, then the enable condition $S_{(i,c)}$ checks whether w admits a factorization of the form $w = w_1 u_{i,j} w_2$ for some index $j \in \{1, \ldots, n_i\}$ such that $¢w_1 \in L(F_{i,j})$ and $w_2\$ \in L(E'_{i,j})$. If such a factorization does not exist, then $P_{(i,c)}$ halts and rejects. However, we see from the discussion above that then none of the meta-instruction $I_{i,j}$ $(1 \leq j \leq n_i)$ of M_i is applicable, either. Otherwise, $w_1 u_{i,j}$ is the shortest prefix of w to which a meta-instruction of M_i applies. Hence, $P_{(i,c)}$ executes exactly the same cycle that M_i would execute in this situation. From the disable condition $T_{(i,c)}$ we see that in mode ed the component $P_{(i,c)}$ will execute a single cycle only. It follows that $L_{\mathsf{ed}}(\mathcal{M}') = L_{=1}(\mathcal{M})$ holds. □

Together with Corollary 8 and the trivial inclusion

$$\mathcal{L}_{\mathsf{ed}}(\text{det-local-CD-RX}) \subseteq \mathcal{L}_{\mathsf{ed}}(\text{det-local-CD-RRX})$$

this yields the following equalities.

Corollary 10. *For all* $\mathsf{X} \in \{\lambda, \mathsf{W}, \mathsf{WW}\}$,

$$\mathcal{L}_{\mathsf{ed}}(\text{det-local-CD-RX}) = \mathcal{L}_{\mathsf{ed}}(\text{det-local-CD-RRX}) = \mathcal{L}_{=1}(\text{det-local-CD-RRX}).$$

7 Concluding Remarks

Is it possible to extend Theorem 6 to globally deterministic CD-RR(W)(W)-systems? Let $\mathcal{M} := ((M_i, \sigma_i)_{i \in I}, I_0)$ be a globally deterministic CD-RR(W)(W)-system, and let $L = L_{=1}(\mathcal{M})$. Each component M_i can be described by a finite sequence of meta-instructions of the form $(I_{i,0}, I_{i,1}, \ldots, I_{i,n_i})$, where $I_{i,0} = (E_{i,0}, \mathsf{Accept})$ is an accepting meta-instruction, and $I_{i,j} = (E_{i,j}, u_{i,j} \to v_{i,j}, E'_{i,j}, \mathsf{Restart}(\alpha_{i,j}))$ $(1 \leq j \leq n_i)$ are rewriting meta-instructions Here $\alpha_{i,j} \in \sigma_i$ is the unique successor of component M_i that is called once meta-instruction $I_{i,j}$ of M_i has been executed successfully. As in the discussion above M_i must determine the shortest prefix w_1 of w such that $¢w_1 u_{i,j} \in L(E_{i,j}) \cdot u_{i,j}$ for some $j \in \{1, \ldots, n_i\}$. However, in contrast to the situation for locally deterministic CD-RR(W)(W)-systems, the suffix w_2 may influence the subsequent

computation of $\mathcal{M}$ in an essential way. Assume that there exist indices $j \neq r$ such that $L(E_{i,j}) \cap L(E_{i,r}) \neq \emptyset$, $u_{i,j} = u_{i,r}$, and $v_{i,j} = v_{i,r}$, and let $w \in \Gamma^*$ have a factorization of the form $w = w_1 u w_2$ such that $\mathrm{¢}w_1 \in L(E_{i,j}) \cap L(E_{i,r})$ and $u = u_{i,j} = u_{i,r}$. Then it depends on the corresponding suffix w_2 whether component $\alpha_{i,j}$ or $\alpha_{i,r}$ is called next. The former happens if $w_2\$ \in L(E'_{i,j})$, and the latter happens if $w_2\$ \in L(E'_{i,r})$ holds.

Now a simulating globally deterministic CD-R(W)(W)-system will have the difficulty that, after detecting the prefix $\mathrm{¢}w_1 u$ and executing the correct rewrite $u = u_{i,j} = u_{i,r} \to v_{i,j} = v_{i,r}$, it must decide whether to simulate component $M_{\alpha_{i,j}}$ or component $M_{\alpha_{i,r}}$ next. Further, as the simulating CD-R(W)(W)-system is to be globally deterministic, there is only a single successor that is associated with the current cycle. Hence, it is not possible to carry the construction from the proof of Theorem 6 over to globally deterministic CD-systems. It remains open whether or not the corresponding result holds.

Finally, we consider the general case of nondeterministic CD-RR(W)(W)-systems. Let $M = (Q, \Sigma, \Gamma, \mathrm{¢}, \$, q_0, k, \delta)$ be a nondeterministic RR(W)(W)-automaton, and let $I_0 = (E_0, \mathsf{Accept})$ and $I_i = (E_i, u_i \to v_i, E'_i)$ $(1 \leq i \leq n)$ be a sequence of meta-instructions that describes the behaviour of M. Here we can assume without loss of generality that $|u_i| = k$ holds for all $i = 1, \ldots, n$. As M is nondeterministic, the above meta-instructions are used as follows. Assume that M is in the restarting configuration $q_0\mathrm{¢}w\$$. Then M scans the tape from left to right until it detects a prefix w_1 of w such that $w_1 = w_3 u_i$ and $\mathrm{¢}w_3 \in L(E_i)$ for some $i \in \{1, \ldots, n\}$. It then rewrites u_i into v_i and checks whether the corresponding suffix w_2 of w satisfies the condition that $w_2\$ \in L(E'_i)$. Observe that there may exist a shorter prefix $w'_1 = w'_3 u_j$ such that $\mathrm{¢}w'_3 \in L(E_j)$ holds, where the corresponding suffix w'_2 may or may not satisfy the condition that $w'_2\$ \in L(E'_j)$. At the same time M checks whether the original tape content $\mathrm{¢}w\$$ belongs to the language $L(E_0)$. If the latter holds, then M may halt and accept; if $w_2\$ \in L(E'_i)$, then M restarts, which yields the restarting configuration $q_0\mathrm{¢}w_3 v_i w_2\$$. Finally, if $w_2\$ \in L(E'_i)$ does not hold, or if no prefix of the above form is found, then this particular computation of M fails.

When trying to simulate M (or a CD-RR(W)(W)-system $\mathcal{M}$) by a CD-R(W)(W)-system $\mathcal{M}'$ working in mode ed, we have the following problem. Even if the enable condition of a component of $\mathcal{M}'$ tells us that the current tape content $\mathrm{¢}w\$$ belongs to the language $L(E_i) \cdot u_i \cdot L(E'_i)$, and if it detects a prefix $\mathrm{¢}w_1 u_i$ of $\mathrm{¢}w$ such that $\mathrm{¢}w_1 \in L(E_i)$ holds, then this does not guarantee that the corresponding suffix $w_2\$$ is in the language $L(E'_i)$. Hence, it is not possible to carry the construction from the proof of Theorem 6 over to nondeterministic CD-systems. It remains open whether or not the corresponding result holds.

Also the following questions and problems remain for future work:

1. Does Theorem 4 extend to CD-R-, CD-RW-, and CD-RWW-automata? If so, then together with Theorem 6 this would imply that also in mode = 1 locally deterministic CD-R(W)(W)-systems are as expressive as locally deterministic CD-RR(W)(W)-systems.

2. Does Theorem 4 extend to other modes of operation? Or is it possible to establish a proper separation result, at least for those types of restarting automata that have no auxiliary symbols?
3. Another topic for research is the question about the number of components that are needed to accept a certain language. From the proofs above it appears that less components may be needed in the ed mode of operation than in the $= 1$ mode of operation.

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 and J. Kelemen. Cooperating/Distributed grammar systems: A link between formal languages and artificial intelligence, *Bulletin of the EATCS* 45 (1991) 131–145.
3. P. Jančar, F. Mráz, M. Plátek, and J. Vogel. Restarting automata. In: H. Reichel (ed.), *FCT 1995, Proc.*, *Lect. Notes Comput. Sci. 965*, Springer, Berlin, 1995, 283–292.
4. H. Messerschmidt and F. Otto. On nonforgetting restarting automata that are deterministic and/or monotone. In: D. Grigoriev, J. Harrison, and E.A. Hirsch (eds.), *CSR 2006, Proc.*, *Lect. Notes Comput. Sci. 3967*, Springer, Berlin, 2006, 247–258.
5. H. Messerschmidt and F. Otto. Cooperating distributed systems of restarting automata. *Int. J. Found. Comput. Sci.* 18 (2007) 1333–1342.
6. 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.
7. H. Messerschmidt and F. Otto. On deterministic CD-systems of restarting automata. *Int. J. Found. Comput. Sci.* 20 (2009) 185–209.
8. H. Messerschmidt and H. Stamer. Restart-Automaten mit mehreren Restart-Zuständen. In: H. Bordihn (ed.), *Workshop "Formale Methoden in der Linguistik" und 14. Theorietag "Automaten und Formale Sprachen", Proc.*, Institut für Informatik, Universität Potsdam, 2004, 111–116.
9. F. Otto. Restarting automata and their relations to the Chomsky hierarchy. In: Z. Esik and Z. Fülöp (eds.), *DLT 2003, Proc.*, *Lect. Notes Comput. Sci. 2710*, Springer, Berlin, 2003, 55-74.
10. F. Otto. Restarting automata. In: Z. Ésik, C. Martin-Vide, and V. Mitrana (eds.), *Recent Advances in Formal Languages and Applications*, Studies in Computational Intelligence Vol. 25, Springer, Berlin, 2006, 269–303.