RAIRO-Theor. Inf. Appl. 45 (2011) 413–448
DOI: 10.1051/ita/2011123

Available online at:
www.rairo-ita.org

CD-SYSTEMS OF STATELESS DETERMINISTIC R(1)-AUTOMATA GOVERNED BY AN EXTERNAL PUSHDOWN STORE*,**

Benedek Nagy[1] and Friedrich Otto[2]

Abstract. We study cooperating distributed systems (CD-systems) of stateless deterministic restarting automata with window size 1 that are equipped with an external pushdown store. In this way we obtain an automata-theoretical characterization for the class of word languages that are linearizations of context-free trace languages.

Mathematics Subject Classification. 68Q45.

1. Introduction

Starting with Mazurkiewicz's seminal paper [18], the theory of traces has become an important part of the theory of concurrent systems. A *trace* is an equivalence class of words over a given alphabet with respect to a partial commutativity relation [6]. Informally speaking, if the letters of an alphabet Σ are interpreted

Keywords and phrases. Restarting automaton, cooperating distributed system, external pushdown, context-free trace language.

* *The results of this paper have been announced at SOFSEM 2011 in Nový Smokovec, Slovakia, January 2011. An extended abstract appeared in the proceedings of that conference [23].*

** *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.*

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

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

Article published by EDP Sciences

as *atomic actions*, then a word w over Σ stands for a finite sequence of such actions. If some of these atomic actions, say a and b, are *independent* of each other, then it does not matter in which order they are executed, that is, the sequences of actions ab and ba yield the same result. If D is a reflexive and symmetric *dependency relation* on Σ, and $I = (\Sigma \times \Sigma) \setminus D$ is the corresponding *independence relation*, then the equivalence relation $\equiv_D$ is induced by the pairs $\{ (xaby, xbay) \mid (a,b) \in I,\, x, y \in \Sigma^* \}$. By collecting all words (sequences) that are equivalent to a given word w into a class $[w]_D = \{ z \in \Sigma^* \mid z \equiv_D w \}$, one abstracts from the order between independent actions. These equivalence classes are called *traces*, and the set $\{ [w]_D \mid w \in \Sigma^* \}$ of all traces is the *trace monoid* $M(D)$. A set of traces $S \subseteq M(D)$ is called a *trace language*, and the set of all words w such that $[w]_D$ belongs to S is called the *linearization* of this trace language. A trace language $S \subseteq M(D)$ is called *recognizable* (*context-free*) if there exists a regular (context-free) language $R \subseteq \Sigma^*$ such that $S = \{ [w]_D \mid w \in R \}$, and it is called *rational* if it is empty or if it can be obtained from singleton sets by a finite number of unions, products, and star operations. However, in contrast to the situation for words (that is, free monoids), the recognizable trace languages are a proper subclass of the rational trace languages (unless $I = \emptyset$). For a detailed presentation of trace theory and a long list of references, see the monograph by Diekert and Rozenberg [9], which serves as our main reference on this topic.

In [26] Zielonka introduced asynchronous automata for accepting recognizable trace languages. Actually he presented a construction that, starting from a regular I-closed word language that is either given through a homomorphism into a finite monoid, or by an I-diamond automaton, yields a deterministic asynchronous automaton for the corresponding trace language. Interestingly, these automata process traces, not linearizations thereof, in this way reflecting the concurrent aspect of traces. In [8] finite automata are studied that work on *multisets*, that is, they accept recognizable sets of traces over a commutative alphabet (that is, $I = (\Sigma \times \Sigma) \setminus \{ (a,a) \mid a \in \Sigma \}$), and this approach has been extended to *multiset pushdown automata* in [15]. However, so far no automata-theoretical characterization has been obtained for rational or context-free trace languages.

Here we make a step into this direction by studying systems of automata that accept *linearizations* of context-free trace languages. In [21] it is shown that linearizations of rational trace languages are accepted by *cooperating distributed systems* (CD-systems) of a particular type of stateless deterministic restarting automata. Restarting automata were invented by Jančar *et al.* to model the linguistic technique of *analysis by reduction* [13] (see also [25]). These automata can be interpreted as generalizations of finite-state acceptors that work in cycles. They have a finite-state control and a read/write window of a fixed size $k \geq 1$ that works on a *flexible* tape. In each cycle they execute a single rewrite operation that strictly shortens the actual tape. CD-systems of restarting automata have been defined in [19], and in [20] various types of deterministic CD-systems of restarting automata have been studied. In such a system the component automata cooperate in processing a given input word: at each moment exactly one component

automaton is active. It executes one or more cycles, depending on the chosen mode of operation, and then another component automaton becomes active. As expected, CD-systems are much more expressive than their component automata themselves. On the other hand, *stateless* restarting automata, that is, restarting automata with only a single state, have been introduced and studied in [16,17]. 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 corresponding counterparts with states.

In [21] CD-systems of stateless deterministic restarting automata that have a read/write window of size 1 only are considered. Working in mode = 1, these systems accept a class of semi-linear languages that properly contains the linearizations of all rational trace languages. In fact, even a characterization of the linearizations of rational trace languages in terms of a particular type of these CD-systems was obtained.

Here we extend these CD-systems by an external pushdown store that is used to determine the successor of the current automaton, in this way obtaining the so-called *pushdown CD-systems* of stateless deterministic R(1)-automata, abbreviated as PD-CD-R(1)-systems. When the active automaton of such a system performs a cycle, then its successor automaton is chosen based on both, the symbol deleted in this cycle and the topmost symbol on the pushdown store. In this process also the pushdown content is modified by either erasing the topmost symbol, or by replacing it by a word of length at most 2. Essentially such a system can be interpreted as a traditional pushdown automaton, in which the operation of reading an input symbol has been replaced by a stateless deterministic R(1)-automaton. Hence, not the first symbol is necessarily read, but some symbol that can be reached by this automaton by moving across a prefix of the current input word. In this way our CD-systems can be interpreted as pushdown automata with *translucent letters*. Analogously, the CD-systems of stateless deterministic restarting automata with window size 1 studied in [21] can be interpreted as *finite-state acceptors with translucent letters* (see [24]). Also other variants of pushdown automata that do not simply read their input sequentially from left to right have been studied before. For example, in [5] pushdown automata are considered that can reverse their input.

We show that the class $\mathcal{L}(\text{PD-CD-R}(1))$ of languages that are accepted by PD-CD-R(1)-systems is a proper subclass of the class of languages with semi-linear Parikh image, and that it includes the linearizations of all context-free trace languages. Actually, from a context-free word language $R \subseteq \Sigma^*$ that is given through a context-free grammar and a dependency relation D on Σ, we construct a PD-CD-R(1)-system for the linearization of the context-free trace language $S = \{ [w]_D \mid w \in R \}$. On the other hand, from a given PD-CD-R(1)-system $\mathcal{M}$, we can extract a pushdown automaton A such that the language $L(A)$ is a sublanguage of $L(\mathcal{M})$ that is letter-equivalent to $L(\mathcal{M})$. Finally, we present a characterization of the class of linearizations of context-free trace languages in terms of a particular type of PD-CD-R(1)-systems. In fact, from a PD-CD-R(1)-system $\mathcal{M}$ of

this type, one can construct a pushdown automaton B such that the language $L(\mathcal{M})$ accepted by $\mathcal{M}$ is the linearization of the context-free trace language $S = \{ [w]_D \mid w \in L(B) \}$.

This paper is structured as follows. In Section 2 we restate in short the definition of the CD-systems of stateless deterministic R(1)-automata and their main properties from [21], and in Section 3, we define the PD-CD-R(1)-systems. We also consider the special case of these CD-systems where the pushdown is a counter (the so-called OC-CD-R(1)-systems), that is, there is only a single pushdown symbol in addition to the bottom marker. We illustrate these definitions by some examples and compare the resulting language classes to each other and to the class CFL of context-free languages, the class OCL of one-counter languages, and the class $\mathcal{L}_{=1}$(stl-det-local-CD-R(1)) of languages that are accepted by CD-systems of stateless deterministic R(1)-automata. Then in Section 4, we study the classes of linearizations of one-counter and context-free trace languages. We will see that our PD-CD-R(1)-systems accept a proper superclass of the linearizations of context-free trace languages, and the OC-CD-R(1)-systems accept a proper superclass of the linearizations of one-counter trace languages, and we present characterizations of these classes of linearizations of trace languages in terms of our CD-systems. Finally, in Section 5 we state some preliminary closure and non-closure results and several open problems. The paper closes with some concluding remarks in Section 6.

Notation. *For a finite alphabet Σ, we use Σ^+ to denote the set of all non-empty words over Σ, while Σ^* denotes the set of all words over Σ including the empty word ε. For a word $w \in \Sigma^*$ and a letter $a \in \Sigma$, $|w|$ denotes the length of w, and $|w|_a$ denotes the a-length of w, that is, the number of occurrences of the letter a in w. Further, w^R denotes the* reversal *(or* mirror image*) of w.*

If $\Sigma = \{a_1, \ldots, a_n\}$, then the corresponding Parikh mapping *is the morphism $\psi : \Sigma^* \to \mathbb{N}^n$ from the set of words over Σ into the set of vectors of dimension n over $\mathbb{N}$ that is defined by mapping a_i to the vector $(\underbrace{0, \ldots, 0}_{i-1}, 1, \underbrace{0, \ldots, 0}_{n-i})$ for all $1 \leq i \leq n$. Two languages $L_1, L_2 \subseteq \Sigma^*$ are called* letter-equivalent *if $\psi(L_1) = \psi(L_2)$ holds. A language $L \subseteq \Sigma^*$ is called* semi-linear *if its Parikh image $\psi(L)$ is a semi-linear subset of $\mathbb{N}^n$, that is, if $\psi(L)$ is the union of finitely many linear subsets of $\mathbb{N}^n$ (see e.g., [11]).*

We use REG, LIN, DCFL, and CFL to denote the classes of regular, linear, deterministic context-free and context-free languages. The monographs [11,12] are our main references on formal language and automata theory.

2. Stateless deterministic R(1)-automata

Stateless types of restarting automata were introduced in [16]. Here we are only interested in the most restricted form of them, the *stateless deterministic* R*-automaton* of window size 1. A *stateless deterministic* R(1)*-automaton* is a one-tape machine that is described by a 5-tuple $M = (\Sigma, ¢, \$, 1, \delta)$, where Σ is a finite

alphabet, the symbols $¢, \$ \notin \Sigma$ serve as markers for the left and right border of the work space, respectively, the size of the *read/write window* is 1, and $\delta : \Sigma \cup \{¢, \$\} \to \{\mathsf{MVR}, \mathsf{Accept}, \varepsilon\}$ is the (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. In addition, we use the notation $\delta(a) = \emptyset$ to express the fact that the function δ is undefined for the symbol a. Some 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 $(\varepsilon, ¢w\$)$ is usually simply written as $¢w\$$. 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 $¢w\$$, the window moves right until a configuration of the form $(¢x, ay\$)$ is reached such that $\delta(a) = \varepsilon$. Now the latter configuration is transformed into the restarting configuration $¢xy\$$. This computation, which is called a *cycle*, is expressed as $w \vdash_M^c xy$. A computation of M now consists of a finite sequence of cycles that is followed by a *tail computation*, which consists of a sequence of move-right operations possibly followed by an accept step. An input word $w \in \Sigma^*$ is *accepted* by M, if the computation of M which starts with the initial configuration $¢w\$$ finishes by executing an accept step. By $L(M)$ we denote the language consisting of all words accepted by M.

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

$$\Sigma_{\mathrm{M}} = \{\, a \in \Sigma \mid \delta(a) = \mathsf{MVR} \,\},\ \Sigma_{\mathrm{A}} = \{\, a \in \Sigma \mid \delta(a) = \mathsf{Accept} \,\},$$
$$\Sigma_{\varepsilon} = \{\, a \in \Sigma \mid \delta(a) = \varepsilon \,\}, \qquad \Sigma_{\emptyset} = \{\, a \in \Sigma \mid \delta(a) = \emptyset \,\}.$$

Thus, Σ_{M} is the set of letters that M just moves across, Σ_{ε} is the set of letters that M deletes, Σ_{A} is the set of letters which cause M to accept, and $\Sigma_{\emptyset}$ is the set of letters on which M will get stuck. It has been shown in [21] that the language $L(M)$ can be characterized as

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

Let $M = (\Sigma, ¢, \$, 1, \delta)$ be a stateless deterministic R(1)-automaton. If $\delta(¢)$ is undefined, then $L(M) = \emptyset$. Define a stateless deterministic R(1)-automaton

$M_- = (\Sigma, ¢, \$, 1, \delta_-)$ by taking $\delta_-(¢) = \delta_-(a) = \mathsf{MVR}$ for all $a \in \Sigma$ and $\delta_-(\$) = \emptyset$. Then M_- scans its tape contents completely and halts (and rejects) on the right delimiter \$, that is, $L(M_-) = \emptyset$. Similarly, if $\delta(¢) = \mathsf{Accept}$, then $L(M) = \Sigma^*$. Define a stateless deterministic $\mathsf{R}(1)$-automaton $M_+ = (\Sigma, ¢, \$, 1, \delta_+)$ by taking $\delta_+(¢) = \delta_+(a) = \mathsf{MVR}$ for all $a \in \Sigma$ and $\delta_+(\$) = \mathsf{Accept}$. Then M_+ scans its tape contents completely and halts (and accepts) on the right delimiter \$, that is, $L(M_+) = \Sigma^*$. Thus, the automaton M is equivalent to M_- (in the first case) or to M_+ (in the second case). Accordingly, we assume in the following that for all stateless deterministic $\mathsf{R}(1)$-automata $M = (\Sigma, ¢, \$, 1, \delta)$ considered, $\delta(¢) = \mathsf{MVR}$ holds.

Cooperating distributed systems of restarting automata were introduced and studied in [19]. Here we only consider *cooperating distributed systems of stateless deterministic* $\mathsf{R}(1)$*-automata* (or stl-det-local-CD-$\mathsf{R}(1)$-systems for short). Such a system consists of a finite collection $\mathcal{M} = ((M_i, \sigma_i)_{i \in I}, I_0)$ of stateless deterministic $\mathsf{R}(1)$-automata $M_i = (\Sigma, ¢, \$, 1, \delta_i)$ $(i \in I)$, *successor relations* $\sigma_i \subseteq I$ $(i \in I)$, and a subset $I_0 \subseteq I$ of *initial indices*. Here it is required that $I_0 \neq \emptyset$, and that $\sigma_i \neq \emptyset$ for all $i \in I$. These systems are called *locally deterministic* in accordance with the notation coined in [20], since their computations are not deterministic as we will see below, although all their component automata M_i $(i \in I)$ are deterministic. Actually, in [21] it is required additionally that $i \notin \sigma_i$ for all $i \in I$, but as we are only interested in mode $= 1$ computations (see below), this requirement is actually irrelevant. In fact, by simply adding a copy for each component automaton M_i $(i \in I)$, we could easily enforce it, but this would just make the systems larger and harder to describe.

The cooperating distributed systems of restarting automata can be seen as an adaptation of the notion of a CD-grammar system *with external control* (see *e.g.* [7]) to restarting automata. Accordingly various modes of operation have been defined and studied for them [19], but here we concentrate on mode $= 1$ computations only.

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 R-automaton M_{i_0} starts the computation with the initial configuration $¢w\$$, 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. 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 $\mathcal{L}_{=1}$(stl-det-local-CD-$\mathsf{R}(1)$) we denote the class of languages that are accepted by mode $= 1$ computations of stl-det-local-CD-$\mathsf{R}(1)$-systems.

Example 2.1. Let $\mathcal{M} = ((M_i, \sigma_i)_{i \in I}, I_0)$, where $I = \{a, b, c, +\}$, $I_0 = \{a\}$, $\sigma_a = \{b\}$, $\sigma_b = \{c\}$, $\sigma_c = \{a, +\}$, $\sigma_+ = \{a\}$, and M_a, M_b, M_c, and M_+ are the stateless

deterministic R(1)-automata that are given by the following transition functions:

$$\begin{array}{llll} M_a: & \delta_a(¢) = \mathsf{MVR}, & \delta_a(b) = \mathsf{MVR}, & \delta_a(c) = \mathsf{MVR}, \quad \delta_a(a) = \varepsilon, \\ M_b: & \delta_b(¢) = \mathsf{MVR}, & \delta_b(a) = \mathsf{MVR}, & \delta_b(c) = \mathsf{MVR}, \quad \delta_b(b) = \varepsilon, \\ M_c: & \delta_c(¢) = \mathsf{MVR}, & \delta_c(a) = \mathsf{MVR}, & \delta_c(b) = \mathsf{MVR}, \quad \delta_c(c) = \varepsilon, \\ M_+: & \delta_+(¢) = \mathsf{MVR}, & \delta_+(\$) = \mathsf{Accept}, & \end{array}$$

and $\delta_a(\$), \delta_b(\$), \delta_c(\$)$ and $\delta_+(a), \delta_+(b), \delta_+(c)$ are undefined.

The automaton M_+ accepts the empty word and rejects (that is, it gets stuck on) all other inputs. The automaton M_a simply deletes the first occurrence of the letter a from its tape, M_b simply deletes the first occurrence of the letter b, and M_c simply deletes the first occurrence of the letter c. Accordingly $L_{=1}(\mathcal{M})$ is the non-context-free language $L_{abc} = \{\, w \in \{a,b,c\}^* \mid |w|_a = |w|_b = |w|_c \geq 1 \,\}$.

In [21] the following result was established.

Proposition 2.2. *Each language $L \in \mathcal{L}_{=1}(\mathsf{stl\text{-}det\text{-}local\text{-}CD\text{-}R(1)})$ contains a regular sublanguage E that is letter-equivalent to L. In fact, a finite-state acceptor for E can be constructed effectively from a* stl-det-local-CD-R(1)*-system for L.*

In particular, it follows that $\mathcal{L}_{=1}(\mathsf{stl\text{-}det\text{-}local\text{-}CD\text{-}R(1)})$ only contains languages that are semi-linear. Let DLIN denote the class of *deterministic linear languages*, which is the class of languages that are accepted by deterministic one-turn pushdown automata. Further, let DOCL and OCL denote the classes of *deterministic one-counter languages* and *one-counter languages*, which are the classes of languages that are accepted by (deterministic) one-counter automata (see below). The language $L = \{\, a^n b^n \mid n \geq 0 \,\}$ is accepted by a deterministic one-turn pushdown automaton as well as by a deterministic one-counter automaton, that is, it belongs to the intersection $\mathsf{DLIN} \cap \mathsf{DOCL}$. However, it does not contain a regular sublanguage that is letter-equivalent to the language itself. Thus, we see from Proposition 2.2 that this language is not accepted by any stl-det-local-CD-R(1)-system working in mode $= 1$. 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, DOCL, OCL, DCFL, and CFL with respect to inclusion.

For technical reasons the following normal form was introduced in [21] for stl-det-local-CD-R(1)-systems.

Definition 2.3. A stl-det-local-CD-R(1)-system $\mathcal{M} = ((M_i, \sigma_i)_{i \in I}, I_0)$ is in *normal form*, if it satisfies the following three conditions for all $i \in I$, where $\Sigma_{\mathrm{M}}^{(i)}, \Sigma_{\varepsilon}^{(i)}, \Sigma_{\mathrm{A}}^{(i)}, \Sigma_{\emptyset}^{(i)}$ is the partitioning of alphabet Σ for the automaton M_i as described above:

(1) $|\Sigma_{\varepsilon}^{(i)}| \leq 1$, (2) $\delta_i(¢) = \mathsf{MVR}$ and $\Sigma_{\mathrm{A}}^{(i)} = \emptyset$, (3) $\Sigma_{\varepsilon}^{(i)} = \emptyset$ iff $\delta_i(\$) = \mathsf{Accept}$.

It is shown in [21] that a given stl-det-local-CD-R(1)-system $\mathcal{M}$ can be converted effectively into a stl-det-local-CD-R(1)-system $\mathcal{M}'$ in normal form such that $L_{=1}(\mathcal{M}') = L_{=1}(\mathcal{M})$. However, the system $\mathcal{M}'$ can have about $|\Sigma| + 1$ times as many component automata as the given system $\mathcal{M}$. In [22] closure properties and algorithmic properties are presented for the language class $\mathcal{L}_{=1}$(stl-det-local-CD-R(1)).

3. CD-systems with an external pushdown store

A *pushdown CD-system of stateless deterministic* R(1)*-automata*, a PD-CD-R(1)-system for short, consists of a CD-system of stateless deterministic R(1)-automata and an external pushdown store. Essentially such a system can be interpreted as a traditional pushdown automaton, in which the operation of reading an input symbol is replaced by a stateless deterministic R(1)-automaton. Hence, not the first symbol is necessarily read, but some symbol that can be reached by this automaton by moving across a prefix of the current input word. Formally, a PD-CD-R(1)-system is defined as a tuple $\mathcal{M} = (I, \Sigma, (M_i, \sigma_i)_{i\in I}, \Gamma, \bot, I_0, \delta)$, where

- I is a finite set of indices,
- Σ is a finite input alphabet,
- for all $i \in I$, M_i is a stateless deterministic R(1)-automaton on Σ, and $\sigma_i \subseteq I$ is a non-empty set of possible successors for M_i,
- Γ is a finite pushdown alphabet,
- $\bot \notin \Gamma$ is the bottom marker of the pushdown store,
- $I_0 \subseteq I$ is the set of initial indices, and
- $\delta : (I \times \Sigma \times (\Gamma \cup \{\bot\})) \to 2^{I\times(\Gamma\cup\{\bot\})^*}$ is the successor relation. For each $i \in I$, $a \in \Sigma$, and $A \in \Gamma$, $\delta(i, a, A)$ is a subset of $\sigma_i \times \Gamma^{\leq 2}$, and $\delta(i, a, \bot)$ is a subset of $\sigma_i \times (\bot \cdot \Gamma^{\leq 2})$. Here $\Gamma^{\leq 2}$ denotes the set of all words over Γ of length at most 2.

A *configuration* of $\mathcal{M}$ is a triple of the form $(i, \omega, \bot\alpha)$, where $i \in I$, $\omega \in (\mathbb{¢} \cdot \Sigma^* \cdot \$) \cup \{\mathsf{Accept}\}$, and $\alpha \in \Gamma^*$. A configuration of the form $(i, \mathbb{¢}w\$, \bot\alpha)$ describes the situation that the component automaton M_i has just been activated, the word $\mathbb{¢}w\$$ is the corresponding restarting configuration of M_i, and the word $\bot\alpha$ is the current content of the pushdown store with the last symbol of α at the top. For $w \in \Sigma^*$, an *initial configuration* of $\mathcal{M}$ on input w has the form $(i_0, \mathbb{¢}w\$, \bot)$ for any $i_0 \in I_0$, and an *accepting configuration* has the form $(i, \mathsf{Accept}, \bot)$ for any $i \in I$.

Recall from the discussion in Section 2 that we assume that each component automaton M_i ($i \in I$) performs a move-right operation on the ¢-symbol. Further, for each $i \in I$, let $\Sigma_{\mathrm{M}}^{(i)}$, $\Sigma_{\varepsilon}^{(i)}$, and $\Sigma_{\mathrm{A}}^{(i)}$ denote the subsets of Σ that correspond to the automaton M_i. Then the *single-step computation relation* $\Rightarrow_{\mathcal{M}}$ that $\mathcal{M}$ induces on the set of configurations is defined by the following three rules, where

$i \in I$, $w \in \Sigma^*$, $\alpha \in \Gamma^*$, and $A \in \Gamma$:

(1) $(i, ¢w\$, \bot\alpha A) \Rightarrow_{\mathcal{M}} (j, ¢w'\$, \bot\alpha\eta)$ if $\exists u \in {\Sigma_{\mathrm{M}}^{(i)}}^*, a \in \Sigma_\varepsilon^{(i)}, v \in \Sigma^*$ such that $w = uav, w' = uv,$ and $(j, \eta) \in \delta(i, a, A)$;

(2) $(i, ¢w\$, \bot) \Rightarrow_{\mathcal{M}} (j, ¢w'\$, \bot\eta)$ if $\exists u \in {\Sigma_{\mathrm{M}}^{(i)}}^*, a \in \Sigma_\varepsilon^{(i)}, v \in \Sigma^*$ such that $w = uav, w' = uv,$ and $(j, \bot\eta) \in \delta(i, a, \bot)$;

(3) $(i, ¢w\$, \bot) \Rightarrow_{\mathcal{M}} (i, \mathsf{Accept}, \bot)$ if $\exists u \in {\Sigma_{\mathrm{M}}^{(i)}}^*, a \in \Sigma_{\mathrm{A}}^{(i)}, v \in \Sigma^*$ such that $w = uav,$ or $w \in {\Sigma_{\mathrm{M}}^{(i)}}^*$ and $\delta_i(\$) = \mathsf{Accept}$.

Notice that the contents of the pushdown store is always a word of the form $\bot\alpha$ for some $\alpha \in \Gamma^*$, that is, the bottom marker $\bot$ cannot be removed from the pushdown store. By $\Rightarrow_{\mathcal{M}}^*$ we denote the *computation relation* of $\mathcal{M}$, which is the reflexive and transitive closure of the relation $\Rightarrow_{\mathcal{M}}$. The language $L(\mathcal{M})$ accepted by $\mathcal{M}$ consists of all words for which $\mathcal{M}$ has an accepting computation, that is,

$$L(\mathcal{M}) = \{\, w \in \Sigma^* \mid \exists i_0 \in I_0\, \exists i \in I : (i_0, ¢w\$, \bot) \Rightarrow_{\mathcal{M}}^* (i, \mathsf{Accept}, \bot) \,\}.$$

Remark 3.1. The system $\mathcal{M}$ accepts if and when both of the following conditions are satisfied: the currently active component automaton M_i executes an accepting tail computation starting from the current restarting configuration $¢w\$$, and the pushdown store just contains the bottom marker $\bot$. One could relax this acceptance condition by just requiring that the currently active component automaton M_i accepts starting from the current restarting configuration. It is not clear whether that would change the class of languages accepted. However, the requirement that the pushdown store must just contain the bottom marker at the end of an accepting computation can be seen as a kind of *normalization*. Observe that the contents of the pushdown store of $\mathcal{M}$ is manipulated only in steps of the form (1) and (2), and that during each step of either of these forms a component automaton of $\mathcal{M}$ executes a cycle, that is, an input letter is being erased. Thus, there is no way that $\mathcal{M}$ can manipulate its pushdown store without reading (that is, deleting) input symbols, that is, if a configuration of the form $(i, \mathsf{Accept}, \bot\alpha)$ were reached for some $\alpha \neq \varepsilon$, then α could not be popped from the pushdown store.

A PD-CD-R(1)-system $\mathcal{M} = (I, \Sigma, (M_i, \sigma_i)_{i \in I}, \Gamma, \bot, I_0, \delta)$ is called a *one-counter CD-system of stateless deterministic* R(1)*-automata*, OC-CD-R(1)-system for short, if $|\Gamma| = 1$, that is, if there is only a single pushdown symbol in addition to the bottom marker $\bot$. By $\mathcal{L}$(PD-CD-R(1)) we denote the class of languages that are accepted by PD-CD-R(1)-systems, and $\mathcal{L}$(OC-CD-R(1)) denotes the class of languages that are accepted by OC-CD-R(1)-systems.

Example 3.2. We consider the language

$$L = \{\, a^n v \mid v \in \{b, c\}^*, |v|_b = |v|_c = n, n \geq 0 \,\}.$$

As $L \cap (a^* \cdot b^* \cdot c^*) = \{\, a^n b^n c^n \mid n \geq 0 \,\}$ is not context-free, we see that L itself is not context-free. Further, there is no regular sublanguage of L that is letter-equivalent

to L. Hence, by Proposition 2.2, L is not accepted by any stl-det-local-CD-R(1)-system, either. However, we claim that L is accepted by the OC-CD-R(1)-system $\mathcal{M} = (I, \Sigma, (M_i, \sigma_i)_{i \in I}, \Gamma, \bot, I_0, \delta)$ that is defined as follows:

- $I = \{a, b, c, +\}$, $I_0 = \{a, +\}$, $\Sigma = \{a, b, c\}$, and $\Gamma = \{C\}$,
- M_a, M_b, M_c, and M_+ are defined by the following transition functions:

$$\begin{array}{lll} (1)\ \delta_a(¢) = \mathsf{MVR}, & (5)\ \delta_b(¢) = \mathsf{MVR}, & (8)\ \delta_c(¢) = \mathsf{MVR}, \\ (2)\ \delta_a(a) = \varepsilon, & (6)\ \delta_b(b) = \varepsilon, & (9)\ \delta_c(c) = \varepsilon, \\ (3)\ \delta_+(¢) = \mathsf{MVR}, & (7)\ \delta_b(c) = \mathsf{MVR}, & (10)\ \delta_c(b) = \mathsf{MVR}, \\ (4)\ \delta_+(\$) = \mathsf{Accept}, & & \end{array}$$

  where $\delta_x(y)$ $(x \in I, y \in \Sigma \cup \{\$\})$ is undefined for all other cases,
- $\sigma_a = \{a, b\}$, $\sigma_b = \{c\}$, $\sigma_c = \{b, +\}$, and $\sigma_+ = \{+\}$, and
- δ is defined as follows:

$$\begin{array}{ll} (1)\ \delta(a, a, \bot) = \{(a, \bot C), (b, \bot C)\}, & (3)\ \delta(b, b, C) = \{(c, C)\}, \\ (2)\ \delta(a, a, C) = \{(a, CC), (b, CC)\}, & (4)\ \delta(c, c, C) = \{(b, \varepsilon), (+, \varepsilon)\}, \end{array}$$

  and for all other tripels, δ yields the empty set.

The component automaton M_+ just accepts the empty word, and it gets stuck on all other words. The component M_a just deletes the first letter, if it is an a, otherwise, it gets stuck. The component M_b reads across c's and deletes the first b it encounters, and analogously, the component M_c reads across b's and deletes the first c it encounters. Thus, we see from the form of the successor sets that $\mathcal{M}$ can only accept certain words of the form $a^m v$ such that $v \in \{b, c\}^*$. However, when M_a deletes an a, then a symbol C is pushed onto the pushdown store, and when M_c deletes a c, then a symbol C is popped from the pushdown store. As M_b and M_c work alternatingly, this means that the same number of b's and c's are deleted. Thus, if M is to accept, then $|v|_b = |v|_c = n$ holds for some $n \geq 0$.

If $m < n$, then after deleting the first m occurrences of b and c, the pushdown store only contains the bottom marker $\bot$, and then $\mathcal{M}$ gets stuck as seen from the definition of δ. On the other hand, if $m > n$, then the pushdown still contains some occurrences of the symbol C when the word $a^m v$ has been erased completely. Hence, in this situation $\mathcal{M}$ does not accept, either. Finally, if $m = n$, then after erasing the last occurrence of c, also the last occurrence of the symbol C is popped from the pushdown store, and then M_+ can accept starting from the configuration $(+, ¢\$, \bot)$. Hence, we see that $L(\mathcal{M}) = L$ holds.

Thus, already the language class $\mathcal{L}$(OC-CD-R(1)) contains a language that is neither context-free nor accepted by any stl-det-local-CD-R(1)-system.

Remark 3.1 (cont.)**.** In the above example we exploit the fact that $\mathcal{M}$ accepts only when its pushdown store contains nothing but the bottom marker. However, for the language L above an OC-CD-R(1)-system can be designed that accepts with an arbitrary contents in its pushdown store.

Next we show that OC-CD-R(1)-systems can simulate all stl-det-local-CD-R(1)-systems.

Proposition 3.3. $\mathcal{L}_{=1}(\mathsf{stl\text{-}det\text{-}local\text{-}CD\text{-}R(1)}) \subsetneq \mathcal{L}(\mathsf{OC\text{-}CD\text{-}R(1)})$.

Proof. Let $\mathcal{M} = ((M_i, \sigma_i)_{i\in I}, I_0)$ be a stl-det-local-CD-R(1)-system, and let $L = L_{=1}(\mathcal{M})$. We obtain a OC-CD-R(1)-system $\mathcal{M}' = (I, \Sigma, (M_i, \sigma_i)_{i\in I}, \emptyset, \bot, I_0, \delta)$, where Σ is the tape alphabet of $\mathcal{M}$, by defining the transition function δ as follows for all $i \in I$:

$$\begin{aligned} &\delta(i, a, \bot) = \{\, (j, \bot) \mid j \in \sigma_i \,\} && \text{for all } a \in \Sigma_\varepsilon^{(i)}, \\ &\delta(i, a, \bot) = \emptyset && \text{for all } a \in \Sigma \smallsetminus \Sigma_\varepsilon^{(i)}. \end{aligned}$$

Then there is a one-to-one correspondence between the accepting computations of $\mathcal{M}$ and the accepting computations of $\mathcal{M}'$. Thus, $L(\mathcal{M}') = L$. This yields the announced inclusion. Its properness follows from the previous example. □

Further, PD-CD-R(1)-systems accept all context-free languages.

Proposition 3.4. $\mathsf{CFL} \subsetneq \mathcal{L}(\mathsf{PD\text{-}CD\text{-}R(1)})$.

Proof. Let $L \subseteq \Sigma^+$ be a context-free language. Then there exists a context-free grammar $G = (V, \Sigma, S, P)$ in quadratic Greibach normal form for L, that is, for each production $(A \to r) \in P$, the right-hand side r is of the form $r = a\alpha$, where $a \in \Sigma$ and $\alpha \in V^{\leq 2}$. In addition, we can assume that the start symbol S does not occur on the right-hand side of any production. Applied to G, the standard construction of a pushdown automaton from a context-free grammar yields a pushdown automaton $\mathcal{A}$ without ε-moves that, given a word $w \in \Sigma^+$ as input, simulates a left-most G-derivation of w from S (see *e.g.* [12]). In analogy to this construction we build a PD-CD-R(1)-system $\mathcal{M} = (I, \Sigma, (M_i, \sigma_i)_{i\in I}, V, \bot, \{S\}, \delta)$, where $I = V \cup \{+\}$, the stateless deterministic R(1)-automata M_A $(A \in V)$ and M_+ are defined as follows, where $a \in \Sigma$:

$$\begin{aligned} &(1)\ \delta_A(¢) = \mathsf{MVR}, \\ &(2)\ \delta_A(a) = \varepsilon, && \text{if there exists } \gamma \in V^{\leq 2} : (A \to a\gamma) \in P, \\ &(3)\ \delta_A(a) = \emptyset, && \text{otherwise}, \\ &(4)\ \delta_A(\$) = \emptyset, \\ &(5)\ \delta_+(¢) = \mathsf{MVR}, \\ &(6)\ \delta_+(a) = \emptyset, \\ &(7)\ \delta_+(\$) = \mathsf{Accept}, \end{aligned}$$

the sets of successors are defined by $\sigma_A = \sigma_+ = I$ for all $A \in V$, and the successor relation δ is defined as follows, where $A \in V$ and $a \in \Sigma$:

$$\begin{aligned} (1)\ \delta(S, a, \bot) = {} & \{\, (+, \bot) \mid (S \to a) \in P \,\} \\ & \cup \{\, (B, \bot B) \mid (S \to aB) \in P \,\} \\ & \cup \{\, (B, \bot CB) \mid (S \to aBC) \in P \,\}, \\ (2)\ \delta(A, a, A) = {} & \{\, (B, \varepsilon) \mid B \in V \smallsetminus \{S\} \text{ and } (A \to a) \in P \,\} \\ & \cup \{\, (+, \varepsilon) \mid (A \to a) \in P \,\} \\ & \cup \{\, (B, B) \mid (A \to aB) \in P \,\} \\ & \cup \{\, (B, CB) \mid (A \to aBC) \in P \,\}, \end{aligned}$$

and δ yields the empty set for all other values.

We claim that $L(\mathcal{M}) = L$ holds. For establishing this equality, we prove the following technical result, where U denotes the set $U = V \setminus \{S\}$, and $\Rightarrow_G^*$ denotes the left-most derivation relation of G.

Claim. For all $w \in \Sigma^+$, all $A \in U$, and all $\alpha \in U^*$,

$$(A, ¢w\$, \bot\alpha A) \Rightarrow_{\mathcal{M}}^* (+, \mathsf{Accept}, \bot) \text{ iff } A\alpha^R \Rightarrow_G^* w.$$

Proof. "$\Rightarrow$": Assume that $(A, ¢w\$, \bot\alpha A) \Rightarrow_{\mathcal{M}}^* (+, \mathsf{Accept}, \bot)$. We proceed by induction on $|w|$. If $|w| = 1$, then $w = a \in \Sigma$. We see from the definition of $\mathcal{M}$ that $\alpha = \varepsilon$, and that the above computation of $\mathcal{M}$ has the form $(A, ¢w\$, \bot\alpha A) = (A, ¢a\$, \bot A) \Rightarrow_{\mathcal{M}} (+, ¢\$, \bot)$. This implies that $(A \to a) \in P$, that is, $A = A\alpha^R \Rightarrow_G a = w$ holds.

If $|w| = n + 1$ for some $n \geq 1$, then $w = aw'$ for some $a \in \Sigma$ and some word w' of length n. It follows that the above computation of $\mathcal{M}$ has the form

$$(A, ¢w\$, \bot\alpha A) = (A, ¢aw'\$, \bot\alpha A) \Rightarrow_{\mathcal{M}} (B, ¢w'\$, \bot\gamma B) \Rightarrow_{\mathcal{M}}^* (+, \mathsf{Accept}, \bot),$$

where $(A \to a) \in P$, and then $\gamma B = \alpha$, or $(A \to aB) \in P$, and then $\gamma = \alpha$, or $(A \to aBC) \in P$, and then $\gamma = \alpha C$. In each case we obtain the derivation $A\alpha^R \Rightarrow_G aB\gamma^R \Rightarrow_G^* aw' = w$ from the induction hypothesis for w'.

"$\Leftarrow$": Assume that $A\alpha^R \Rightarrow_G^* w$ holds. Again, we proceed by induction on $|w|$. If $|w| = 1$, then $w = a \in \Sigma$, and it follows from the form of the rules of G that $\alpha = \varepsilon$. Thus, $(A \to a) \in P$, and hence,

$$(A, ¢w\$, \bot\alpha A) = (A, ¢a\$, \bot A) \Rightarrow_{\mathcal{M}} (+, ¢\$, \bot) \Rightarrow_{\mathcal{M}} (+, \mathsf{Accept}, \bot)$$

follows. If $|w| = n + 1$ for some $n \geq 1$, then $w = aw'$ for some $a \in \Sigma$ and some word w' of length n. It follows that the above derivation has the form $A\alpha^R \Rightarrow_G aB\gamma^R \Rightarrow_G^* aw' = w$, where in the first step, $(A \to a) \in P$ is used, and then $\alpha = \gamma B$, or $(A \to aB) \in P$ is used, and then $\alpha = \gamma$, or $(A \to aBC) \in P$ is used, and then $\alpha C = \gamma$. In each case we obtain the following computation from the induction hypothesis for w':

$$(A, ¢w\$, \bot\alpha A) = (A, ¢aw'\$, \bot\alpha A) \Rightarrow_{\mathcal{M}} (B, ¢w'\$, \bot\gamma B) \Rightarrow_{\mathcal{M}}^* (+, \mathsf{Accept}, \bot). \quad \square$$

Let $a \in \Sigma$. Then $a \in L$ iff $(S \to a) \in P$ iff $(S, ¢a\$, \bot) \Rightarrow_{\mathcal{M}} (+, ¢\$, \bot) \Rightarrow_{\mathcal{M}} (+, \mathsf{Accept}, \bot)$ iff $a \in L(\mathcal{M})$. Finally, for all $w \in \Sigma^+$ and all $a \in \Sigma$,

$$\begin{aligned} aw \in L &\text{ iff } S \Rightarrow_G aA\alpha^R \Rightarrow_G^* aw \\ &\text{iff } (S \to aA\alpha^R) \in P \text{ and } A\alpha^R \Rightarrow_G^* w \\ &\text{iff } (S, ¢aw\$, \bot) \Rightarrow_{\mathcal{M}} (A, ¢w\$, \bot\alpha A) \Rightarrow_{\mathcal{M}}^* (+, \mathsf{Accept}, \bot). \end{aligned}$$

Hence, it follows that $L(\mathcal{M}) = L$.

If the given context-free language includes the empty word, we can apply the above construction to the language $L \setminus \{\varepsilon\}$. Then the resulting PD-CD-R(1)-system will accept this language. By adding the component $+$ to the set of initial components, we obtain a PD-CD-R(1)-system for the language L. This yields the intended inclusion, which is proper by Example 3.2. $\square$

Next we consider the so-called *one-counter automata* and the class of languages accepted by them. One finds several different non-equivalent definitions for one-counter automata in the literature. Here we take a definition that is equivalent to the one used by Jančar *et al.* in [14] (see also [3]).

A pushdown automaton $\mathcal{A} = (Q, \Sigma, \Gamma, q_0, \bot, \delta, F)$ is called a *one-counter automaton* if $|\Gamma| = 1$, and if the bottom marker $\bot$ cannot be removed from the pushdown store. Thus, if C is the only symbol in Γ, then the pushdown contents $\bot C^m$ can be interpreted as the integer m for all $m \geq 0$. Accordingly, the pop operation can be interpreted as the decrement -1. It can be assumed in addition that the only other pushdown operations leave the value m unchanged or increase it by 1, that is, the pushdown is not changed or exactly one additional C is pushed onto it. Finally, $\mathcal{A}$ has to read an input symbol in each step, that is, it cannot make any ε-steps.

A word $w \in \Sigma^*$ is accepted by $\mathcal{A}$, if $(q_0, w, \bot) \vdash_{\mathcal{A}}^* (q, \varepsilon, \bot)$ holds for some final state $q \in F$. Observe that $\mathcal{A}$ can only distinguish between two states of its pushdown store: either the topmost symbol is C, which is interpreted by saying that *the counter is positive*, or it is the bottom marker $\bot$, which is interpreted as *the counter is zero*. By OCL we denote the class of languages accepted by one-counter automata. It is well-known that $\mathsf{REG} \subsetneq \mathsf{OCL} \subsetneq \mathsf{CFL}$ holds (see *e.g.* [3]).

Proposition 3.5. $\mathsf{OCL} \subsetneq \mathcal{L}(\mathsf{OC\text{-}CD\text{-}R(1)})$.

Proof. Let $\mathcal{A} = (Q, \Sigma, \{C\}, q_0, \bot, \delta_{\mathcal{A}}, F)$ be a *one-counter automaton*, and let $L = L(\mathcal{A}) \subseteq \Sigma^*$ be the language it accepts. We simulate $\mathcal{A}$ through a $\mathsf{OC\text{-}CD\text{-}R(1)}$-system $\mathcal{M} = (I, \Sigma, (M_i, \sigma_i)_{i \in I}, \{C\}, \bot, I_0, \delta)$, where $I = (Q \times \{=, >\}) \cup \{+\}$, $I_0 = \{(q_0, =), +\}$, $\sigma_{(q,>)} = \sigma_{(q,=)} = \sigma_+ = I$ for all $q \in Q$, the stateless deterministic $\mathsf{R(1)}$-automata $M_{(q,>)}$, $M_{(q,=)}$ $(q \in Q)$, and M_+ are defined as follows:

$$
\begin{aligned}
&(1)\ \delta_{(q,=)}(\text{¢}) = \mathsf{MVR}, \\
&(2)\ \delta_{(q,=)}(a) = \varepsilon, && \text{if } \delta_{\mathcal{A}}(q, a, \bot) \text{ is defined}, \\
&(3)\ \delta_{(q,=)}(a) = \emptyset, && \text{otherwise}, \\
&(4)\ \delta_{(q,=)}(\$) = \emptyset, \\
&(5)\ \delta_{(q,>)}(\text{¢}) = \mathsf{MVR}, \\
&(6)\ \delta_{(q,>)}(a) = \varepsilon, && \text{if } \delta_{\mathcal{A}}(q, a, C) \text{ is defined}, \\
&(7)\ \delta_{(q,>)}(a) = \emptyset, && \text{otherwise}, \\
&(8)\ \delta_{(q,>)}(\$) = \emptyset, \\
&(9)\ \delta_+(\text{¢}) = \mathsf{MVR}, \\
&(10)\ \delta_+(a) = \emptyset, \\
&(11)\ \delta_+(\$) = \mathsf{Accept},
\end{aligned}
$$

and the successor relation δ is defined as follows, where $q \in Q$, $a \in \Sigma$, and $i \in \{1,2\}$:

$$\begin{aligned}
(1)\ \delta((q,=),a,\bot) &= \{\,((q',=),\bot) \mid (q',\bot) \in \delta_{\mathcal{A}}(q,a,\bot)\,\} \\
&\quad \cup \{\,(+,\bot) \mid \exists q' \in F : (q',\bot) \in \delta_{\mathcal{A}}(q,a,\bot)\,\} \\
&\quad \cup \{\,((q',>),\bot C) \mid (q',\bot C) \in \delta_{\mathcal{A}}(q,a,\bot)\,\}, \\
(2)\ \delta((q,>),a,C) &= \{\,((q',>),C^i) \mid (q',C^i) \in \delta_{\mathcal{A}}(q,a,C)\,\} \\
&\quad \cup \{\,((q',>),\varepsilon),((q',=),\varepsilon) \mid (q',\varepsilon) \in \delta_{\mathcal{A}}(q,a,C)\,\} \\
&\quad \cup \{\,(+,\varepsilon) \mid \exists q' \in F : (q',\varepsilon) \in \delta_{\mathcal{A}}(q,a,C)\,\},
\end{aligned}$$

while δ yields the empty set for all other values.

Observe that each time $\mathcal{A}$ decreases its counter, $\mathcal{M}$ also decreases its counter, and in addition it has the option of activating the final component M_+, if the state entered is final. However, M_+ can only accept, if at that moment the input has been processed completely, and $\mathcal{M}$ only accepts if, in addition, the counter is zero. It follows that there is a one-to-one correspondence between the accepting computations of the one-counter automaton $\mathcal{A}$ and the system $\mathcal{M}$. Hence, we have $L(\mathcal{M}) = L(\mathcal{A}) = L$. This yields the intended inclusion, which is proper by Example 3.2. □

Definition 3.6. A PD-CD-R(1)-system $\mathcal{M} = (I, \Sigma, (M_i, \sigma_i)_{i\in I}, \Gamma, \bot, I_0, \delta)$ is in *strong normal form* if it satisfies the following conditions, where, for all $i \in I$, $\Sigma_{\mathrm{M}}^{(i)}, \Sigma_{\varepsilon}^{(i)}, \Sigma_{\mathrm{A}}^{(i)}, \Sigma_{\emptyset}^{(i)}$ is the partitioning of alphabet Σ for the automaton M_i as described in Section 2:

(1) $\exists\, i_+ \in I : \delta_{i_+}(\mathrm{¢}) = \mathsf{MVR}, \delta_{i_+}(\$) = \mathsf{Accept}$, and $\Sigma_{\emptyset}^{(i_+)} = \Sigma$;
(2) $\forall i \in I \setminus \{i_+\} : \delta_i(\mathrm{¢}) = \mathsf{MVR}, |\Sigma_{\varepsilon}^{(i)}| = 1, \Sigma_{\mathrm{A}}^{(i)} = \emptyset$, and $\delta_i(\$) = \emptyset$.

Thus, if $\mathcal{M}$ is in strong normal form, then it has a unique component M_{i_+} that can execute accept instructions, but it only accepts the empty word, while all other components each delete a single kind of letter. In particular, a word $w \in L(\mathcal{M})$ is first erased completely by executing $|w|$ many cycles, and then the empty word is accepted by activating component M_{i_+}. As OC-CD-R(1)-systems are a special type of PD-CD-R(1)-systems, this definition also applies to them. The following technical result shows that we can restrict our attention to PD-CD-R(1)-systems in strong normal form.

Lemma 3.7. *From a* PD-CD-R(1)*-system* $\mathcal{M}$ *one can construct a* PD-CD-R(1)*-system* $\mathcal{M}'$ *in strong normal form such that* $L(\mathcal{M}') = L(\mathcal{M})$*. In addition, if* $\mathcal{M}$ *is an* OC-CD-R(1)*-system, then* $\mathcal{M}'$ *can be constructed to be an* OC-CD-R(1)*-system, too.*

Proof. Let $\mathcal{M} = (I, \Sigma, (M_i, \sigma_i)_{i\in I}, \Gamma, \bot, I_0, \delta)$ be a PD-CD-R(1)-system. As pointed out in Section 2, we can assume that each of the component automata M_i executes

a move-right step on seeing the ¢-symbol, that is, $\delta_i(¢) = \mathsf{MVR}$ for all $i \in I$, where δ_i denotes the transition function of M_i.

First we split every component automaton M_i into $|\Sigma_\varepsilon^{(i)}| + 1$ many parts, $M_i^{(a)}$ for $a \in \Sigma_\varepsilon^{(i)}$, and $M_i^{(+)}$, where the former is responsible for executing the cycles of M_i in which an occurrence of the letter a is deleted, while the latter takes care of the accepting tail computations of M_i. In detail, for each $a \in \Sigma_\varepsilon^{(i)}$,

$$\begin{array}{llll}
\delta_i^{(a)}(¢) = \mathsf{MVR} & \text{and} & \delta_i^{(+)}(¢) = \mathsf{MVR}, & \\
\delta_i^{(a)}(a) = \varepsilon & \text{and} & \delta_i^{(+)}(a) = \emptyset, & \\
\delta_i^{(a)}(b) = \mathsf{MVR} & \text{and} & \delta_i^{(+)}(b) = \mathsf{MVR} & \text{for all } b \in \Sigma_{\mathrm{M}}^{(i)}, \\
\delta_i^{(a)}(b) = \emptyset & \text{and} & \delta_i^{(+)}(b) = \mathsf{Accept} & \text{for all } b \in \Sigma_{\mathrm{A}}^{(i)}, \\
\delta_i^{(a)}(b) = \emptyset & \text{and} & \delta_i^{(+)}(b) = \emptyset & \text{for all } b \in (\Sigma_\varepsilon^{(i)} \smallsetminus \{a\}) \cup \Sigma_\emptyset^{(i)}, \\
\delta_i^{(a)}(\$) = \emptyset, & \text{and} & \delta_i^{(+)}(\$) = \delta_i(\$). &
\end{array}$$

We adjust the successor relations σ_i $(i \in I)$ as

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

and we take

$$\hat{\mathcal{M}} = (\hat{I}, \Sigma, (M_i^{(a)}, \sigma_i^{(a)})_{i \in I, a \in \Sigma_\varepsilon^{(i)}} \cup (M_i^{(+)}, \sigma_i^{(+)})_{i \in I}, \Gamma, \bot, \hat{I}_0, \hat{\delta}),$$

where $\hat{I} = \{\, i^{(a)}, i^{(+)} \mid i \in I,\, a \in \Sigma_\varepsilon^{(i)} \,\}$ and $\hat{I}_0 = \{\, i^{(a)}, i^{(+)} \mid i \in I_0,\, a \in \Sigma_\varepsilon^{(i)} \,\}$. Finally, the successor relation $\hat{\delta} : (\hat{I} \times \Sigma \times (\Gamma \cup \{\bot\}) \to 2^{\hat{I} \times (\Gamma \cup \{\bot\})^*}$ is defined as follows, where $i \in I$, $a, b \in \Sigma$, and $A \in \Gamma$:

$$\begin{array}{ll}
(1)\ \hat{\delta}(i^{(a)}, a, A) = & \{\, (j^{(c)}, \alpha) \mid (j, \alpha) \in \delta(i, a, A), c \in \Sigma_\varepsilon^{(j)} \,\} \\
& \cup \{\, (j^{(+)}, \varepsilon) \mid (j, \varepsilon) \in \delta(i, a, A) \,\}, \\
(2)\ \hat{\delta}(i^{(a)}, a, \bot) = & \{\, (j^{(c)}, \bot\alpha) \mid (j, \bot\alpha) \in \delta(i, a, \bot), c \in \Sigma_\varepsilon^{(j)} \,\} \\
& \cup \{\, (j^{(+)}, \bot) \mid (j, \bot) \in \delta(j, a, \bot) \,\},
\end{array}$$

and for all other tripels, $\hat{\delta}$ yields the empty set.

Then $\hat{\mathcal{M}}$ simply simulates the computations of $\mathcal{M}$. Each time a successor automaton M_j is chosen in a computation of $\mathcal{M}$, one has to guess whether another cycle will be executed, and if so, which rewrite instruction will be applied, or whether the next component automaton will accept in a tail computation. Then in the simulating computation of $\hat{\mathcal{M}}$, one must simply choose the corresponding component $M_j^{(a)}$ or $M_j^{(+)}$. Observe that a computation of $\mathcal{M}$ can succeed only if the pushdown contents just consists of the bottom marker $\bot$ at the moment when the active component M_j executes an accepting tail computation. Accordingly $\hat{\mathcal{M}}$ only needs to be able to choose an accepting component $M_j^{(+)}$ when the pushdown content is just $\bot$ (see (2)) or when it could just now have been reduced to $\bot$ (see (1)). It follows easily that $L(\hat{\mathcal{M}}) = L(\mathcal{M})$.

We now construct the intended system in strong normal form by modifying the system $\hat{\mathcal{M}}$. Observe that all component automata $M_i^{(a)}$ of $\hat{\mathcal{M}}$ already satisfy

the conditions stated in part (2) of Definition 3.6. Hence, it remains to modify the accepting component automata $M_i^{(+)}$ $(i \in I)$. First we introduce a special component M'_+ that just accepts the empty word, that is, $\delta'_+(¢) = \mathsf{MVR}$, $\delta'_+(\$) = \mathsf{Accept}$, and $\delta'_+(a) = \emptyset$ for all letters $a \in \Sigma$. Now we need to distinguish two cases.

Case 1: $\delta_i^{(+)}(\$) = \emptyset$. In this situation $M_i^{(+)}$ accepts all words from the set $\Sigma_{\mathrm{M}}^{(i)^*} \cdot \Sigma_{\mathrm{A}}^{(i)} \cdot \Sigma^*$. We now replace $M_i^{(+)}$ by the two components $M_i'^{(+)}$ and $M_i''^{(+)}$ that are defined by the following transition functions:

$$\begin{array}{lll}
\delta_i'^{(+)}(¢) = \mathsf{MVR}, & & \delta_i''^{(+)}(¢) = \mathsf{MVR}, \\
\delta_i'^{(+)}(a) = \mathsf{MVR} & \text{for all } a \in \Sigma_{\mathrm{M}}^{(i)}, & \delta_i''^{(+)}(a) = \varepsilon \quad \text{for all } a \in \Sigma, \\
\delta_i'^{(+)}(a) = \emptyset & \text{for all } a \in \Sigma_{\varepsilon}^{(i)} \cup \Sigma_{\emptyset}^{(i)}, & \delta_i''^{(+)}(\$) = \emptyset, \\
\delta_i'^{(+)}(a) = \varepsilon & \text{for all } a \in \Sigma_{\mathrm{A}}^{(i)}, & \\
\delta_i'^{(+)}(\$) = \emptyset. & &
\end{array}$$

Further, the successor relation $\hat{\delta}$ is modified as follows:

(1) On the right-hand side of $\hat{\delta}$, each occurrence of the component $M_i^{(+)}$ is replaced by the component $M_i'^{(+)}$.
(2) The following transitions are added:

$$\begin{aligned}
\hat{\delta}(i'^{(+)}, a, \bot) &= \{(i''^{(+)}, \bot), (+', \bot)\} \quad \text{for all } a \in \Sigma_{\mathrm{A}}^{(i)}, \\
\hat{\delta}(i''^{(+)}, a, \bot) &= \{(i''^{(+)}, \bot), (+', \bot)\} \quad \text{for all } a \in \Sigma,
\end{aligned}$$

where $+'$ refers to the component automaton M'_+ introduced above.

This modification of $\hat{\delta}$ ensures that in combination with M'_+, the component automata $M_i'^{(+)}$ and $M_i''^{(+)}$ accept the words from the set $\Sigma_{\mathrm{M}}^{(i)^*} \cdot \Sigma_{\mathrm{A}}^{(i)} \cdot \Sigma^*$.

Case 2: $\delta_i^{(+)}(\$) = \mathsf{Accept}$. In this situation $M_i^{(+)}$ accepts all words from the set $\Sigma_{\mathrm{M}}^{(i)^*} \cdot \Sigma_{\mathrm{A}}^{(i)} \cdot \Sigma^* \cup \Sigma_{\mathrm{M}}^{(i)^*}$. We now replace $M_i^{(+)}$ by the two components $M_i'^{(+)}$ and $M_i''^{(+)}$ defined above and an additional component $\hat{M}_i^{(+)}$ that is defined as follows:

$$\begin{array}{ll}
\hat{\delta}_i^{(+)}(¢) = \mathsf{MVR}, & \hat{\delta}_i^{(+)}(a) = \varepsilon \text{ for all } a \in \Sigma_{\mathrm{M}}^{(i)}, \\
\hat{\delta}_i^{(+)}(\$) = \emptyset, & \hat{\delta}_i^{(+)}(a) = \emptyset \text{ for all } a \in \Sigma \setminus \Sigma_{\mathrm{M}}^{(i)}.
\end{array}$$

In this case the successor relation $\hat{\delta}$ is modified as follows:

(1) On the right-hand side of $\hat{\delta}$, each occurrence of the component $M_i^{(+)}$ is replaced by the components $M_i'^{(+)}$ and $\hat{M}_i^{(+)}$.
(2) The following transitions are added:

$$\begin{aligned}
\hat{\delta}(i'^{(+)}, a, \bot) &= \{(i''^{(+)}, \bot), (+', \bot)\} \quad \text{for all } a \in \Sigma_{\mathrm{A}}^{(i)}, \\
\hat{\delta}(i''^{(+)}, a, \bot) &= \{(i''^{(+)}, \bot), (+', \bot)\} \quad \text{for all } a \in \Sigma, \\
\delta'(\hat{i}^{(+)}, a, \bot) &= \{(\hat{i}^{(+)}, \bot), (+', \bot)\} \quad \text{for all } a \in \Sigma_{\mathrm{M}}^{(i)}.
\end{aligned}$$

This modification of $\hat{\delta}$ ensures that in combination with M'_+, the automata $M_i'^{(+)}$, $M_i''^{(+)}$ and $\hat{M}_i^{(+)}$ accept the words from the set ${\Sigma_{\mathrm{M}}^{(i)}}^* \cdot \Sigma_{\mathrm{A}}^{(i)} \cdot \Sigma^* \cup {\Sigma_{\mathrm{M}}^{(i)}}^*$.

In the revised system $\hat{\mathcal{M}}$, the component M'_+ is the only one that executes accept instructions, and it satisfies the conditions in part (1) of Definition 3.6. In order to obtain the intended system $\mathcal{M}'$ in strong normal form it remains to split each of the component automaton $M_i'^{(+)}$, $M_i''^{(+)}$ and $\hat{M}_i^{(+)}$ that contains more than one rewrite instruction into several automata, one for each letter that can be deleted. Then the resulting PD-CD-R(1)-system $\mathcal{M}'$ is in strong normal form, and it accepts the same language as the original system $\mathcal{M}$.

From the description above we see that the PD-CD-R(1)-system $\mathcal{M}'$ is actually a OC-CD-R(1)-system, if the given system $\mathcal{M}$ is. This completes the proof of Lemma 3.7. □

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

Theorem 3.8. *Each language $L \in \mathcal{L}$(PD-CD-R(1)) contains a context-free sub-language E such that $\psi(L) = \psi(E)$ holds. In fact, a pushdown automaton for E can be constructed effectively from a* PD-CD-R(1)*-system for L.*

Proof. Let $\mathcal{M} = (I, \Sigma, (M_i, \sigma_i)_{i\in I}, \Gamma, \bot, I_0, \delta)$ be a PD-CD-R(1)-system, and let $L = L(\mathcal{M})$. By Lemma 3.7 we can assume that $\mathcal{M}$ is in strong normal form, that is, there exists a unique index $+ \in I$ such that M_+ accepts the empty word, and for each other index $i \in I_r = I \setminus \{+\}$, M_i does not execute any accept instructions and $|\Sigma_\varepsilon^{(i)}| = 1$. To simplify the notation in the following we denote the letter $a \in \Sigma_\varepsilon^{(i)}$ simply by $a_{(i)}$.

From $\mathcal{M}$ we construct a pushdown automaton P. Essentially P is obtained from $\mathcal{M}$ by ignoring all move-right operations on letters, that is, each component automaton M_i $(i \in I)$ is simply modelled by a state of P. Formally, $P = (Q, \Sigma, \Gamma, q_0, \bot, \delta_P, F)$ is defined as follows:

- $Q = I \cup \{q_0\}$, where q_0 is a new state, $F = \{+\}$, and
- the transition relation δ_P is defined by

$$\begin{aligned}
&(1)\ \delta_P(q_0, \varepsilon, \bot) = \{\, (i, \bot) \mid i \in I_0 \,\},\\
&(2)\ \delta_P(i, a, \bot) \ = \{\, (j, \bot\eta) \mid (j, \bot\eta) \in \delta(i, a, \bot) \,\},\\
&(3)\ \delta_P(i, a, A) \ = \{\, (j, \alpha) \mid (j, \alpha) \in \delta(i, a, A) \,\},
\end{aligned}$$

where $i \in I_r$, $a \in \Sigma$, and $A \in \Gamma$, and δ_P is undefined for all other cases.

Then $E = L(P)$ is a context-free language. It remains to show that it is a sub-language of L that is letter-equivalent to L. Observe that $\varepsilon \in L$ holds if and only if $+ \in I_0$ if and only if $(+, \bot) \in \delta_P(q_0, \varepsilon, \bot)$ if and only if $\varepsilon \in E$. Thus, in the following we only need to consider nonempty words.

Claim 1. For all $i_0 \in I$, all $w \in \Sigma^+$, and all $\alpha \in \Gamma^*$, if $(i_0, ¢w\$, \bot\alpha) \Rightarrow_{\mathcal{M}} (i_1, ¢w_1\$, \bot\alpha_1) \Rightarrow_{\mathcal{M}} \cdots \Rightarrow_{\mathcal{M}} (i_s, ¢w_s\$, \bot\alpha_s) \Rightarrow_{\mathcal{M}} (+, ¢\$, \bot) \Rightarrow_{\mathcal{M}} (+, \mathsf{Accept}, \bot)$ is an accepting computation of $\mathcal{M}$, then there exists a word $z \in \Sigma^+$ such that $(i_0, z, \bot\alpha) \vdash_P^* (+, \varepsilon, \bot)$ holds, and $\psi(z) = \psi(w)$.

Proof. We proceed by induction on s. If $s = 0$, then $i_0 \in I_r$, $w = a_{(i_0)}$, $|\alpha| \leq 1$, and $(+, \bot) \in \delta(i_0, a_{(i_0)}, \bot)$, if $\alpha = \varepsilon$, or $(+, \varepsilon) \in \delta(i_0, a_{(i_0)}, A)$, if $\alpha = A \in \Gamma$. If we take $z = a_{(i_0)}$, then $(i_0, z, \bot\alpha) = (i_0, a_{(i_0)}, \bot\alpha) \vdash_P (+, \varepsilon, \bot)$ by (2) or (3).

Now assume that $s \geq 1$. By the induction hypothesis there exists a word $z_1 \in \Sigma^+$ such that $(i_1, z_1, \bot\alpha_1) \vdash_P^* (+, \varepsilon, \bot)$ and z_1 is letter-equivalent to w_1. As $(i_0, ¢w\$, \bot\alpha) \Rightarrow_{\mathcal{M}} (i_1, ¢w_1\$, \bot\alpha_1)$, w has a factorization $w = uav$ such that $u \in \Sigma_{\mathrm{M}}^{(i_0)^*}$, $a = a_{(i_0)}$, and $w_1 = uv$. Further, $\alpha = \varepsilon$ and $(i_1, \bot\alpha_1) \in \delta(i_0, a_{(i_0)}, \bot)$, or $\alpha = \alpha' A$ for some $A \in \Gamma$, and $\alpha_1 = \alpha'\gamma$ such that $(i_1, \gamma) \in \delta(i_1, a_{(i_0)}, A)$.

We define $z = a_{(i_0)} z_1$. Then z is letter-equivalent to $a_{(i_0)} w_1$ and therewith to w, and $(i_0, z, \bot\alpha) = (i_0, a_{(i_0)} z_1, \bot\alpha) \vdash_P (i_1, z_1, \bot\alpha_1) \vdash_P^* (+, \varepsilon, \bot)$. This completes the proof of Claim 1. □

If $w \in L(\mathcal{M})$, then there exists an initial index $i_0 \in I_0$ such that

$$(i_0, ¢w\$, \bot) \Rightarrow_{\mathcal{M}}^* (+, ¢\$, \bot) \Rightarrow_{\mathcal{M}} (+, \mathsf{Accept}, \bot)$$

holds. If $w \neq \varepsilon$, then it follows from Claim 1 that there exists a word z that is letter-equivalent to w such that $(i_0, z, \bot) \vdash_P^* (+, \varepsilon, \bot)$. Hence, we obtain that $z \in L(P)$, as $(q_0, z, \bot) \vdash_P (i_0, z, \bot)$ by (1). Thus, for each $w \in L$, there exists a word $z \in E$ that is letter-equivalent to w.

Claim 2. For all $i \in I$, all $z \in \Sigma^*$, and all $\alpha \in \Gamma^*$, if $(i, z, \bot\alpha) \vdash_P^* (+, \varepsilon, \bot)$, then also $(i, ¢z\$, \bot\alpha) \Rightarrow_{\mathcal{M}}^* (+, ¢\$, \bot) \Rightarrow_{\mathcal{M}} (+, \mathsf{Accept}, \bot)$ holds.

Proof. We proceed by induction on the length n of the computation $(i, z, \bot\alpha) \vdash_P^* (+, \varepsilon, \bot)$. If $n = 0$, then $i = +$, $z = \varepsilon$ and $\alpha = \varepsilon$, and hence, $(i, ¢z\$, \bot\alpha) = (+, ¢\$, \bot) \Rightarrow_{\mathcal{M}} (+, \mathsf{Accept}, \bot)$ holds.

If $n \geq 1$, then $i \in I_r$ and $z = az'$ for some $a \in \Sigma$ and $z' \in \Sigma^*$, and the above computation of P has the following form:

$$(i, z, \bot\alpha) = (i, az', \bot\alpha) \vdash_P (i_1, z', \bot\alpha_1) \vdash_P^* (+, \varepsilon, \bot)$$

for some $i_1 \in I$ and $\alpha_1 \in \Gamma^*$. From the definition of P we see that $a = a_{(i)}$ and either $\alpha = \varepsilon$ and $(i_1, \bot\alpha_1) \in \delta(i, a, \bot)$, or $\alpha = \gamma A$ for some $\gamma \in \Gamma^*$ and $A \in \Gamma$, $\alpha_1 = \gamma\eta$ for some $\eta \in \Gamma^*$, and $(i_1, \eta) \in \delta(i, a, A)$. Thus, $\mathcal{M}$ can perform the computational step $(i, ¢z\$, \bot\alpha) = (i, ¢az'\$, \bot\alpha) \Rightarrow_{\mathcal{M}} (i_1, ¢z'\$, \bot\alpha_1)$. As we know from the induction hypothesis that $\mathcal{M}$ has a computation of the form

$$(i_1, ¢z'\$, \bot\alpha_1) \Rightarrow_{\mathcal{M}}^* (+, ¢\$, \bot) \Rightarrow_{\mathcal{M}} (+, \mathsf{Accept}, \bot),$$

this completes the proof of Claim 2. □

Let $z \in \Sigma^+$. If $z \in E$, then P has an accepting computation of the form $(q_0, z, \bot) \vdash_P (i, z\bot) \vdash_P^* (+, \varepsilon, \bot)$ for some index $i \in I_0$. Hence, we see from Claim 2 that $\mathcal{M}$ can execute the accepting computation $(i, ¢z\$, \bot\alpha) \Rightarrow_{\mathcal{M}}^* (+, ¢\$, \bot) \Rightarrow_{\mathcal{M}} (+, \mathsf{Accept}, \bot)$, which shows that $z \in L$ holds. Hence, it follows that E is indeed a subset of L. This completes the proof of Theorem 3.8. □

In the proof of Theorem 3.8 the pushdown automaton P constructed from the given PD-CD-R(1)-system $\mathcal{M}$ can easily be turned into a one-counter automaton if $\mathcal{M}$ is an OC-CD-R(1)-system. Thus, we also have the following result.

Corollary 3.9. *Each language $L \in \mathcal{L}$(OC-CD-R(1)) contains a sublanguage E that is a one-counter language such that $\psi(L) = \psi(E)$ holds. In fact, a one-counter automaton for E can be constructed effectively from an* OC-CD-R(1)*-system for L.*

As each context-free language has a semi-linear Parikh image, Theorem 3.8 has the following consequence.

Corollary 3.10. *The language class $\mathcal{L}$(PD-CD-R(1)) only contains semi-linear languages, that is, if a language L over $\Sigma = \{a_1, \ldots, a_n\}$ is accepted by a* PD-CD-R(1)*-system, then its Parikh image $\psi(L)$ is a semi-linear subset of $\mathbb{N}^n$.*

The semi-linear language $L = \{ a^n b^n c^n \mid n \geq 0 \}$ does not contain a context-free sublanguage that is letter-equivalent to the language itself. Hence, Theorem 3.8 yields the following negative result.

Proposition 3.11. *The language $L = \{ a^n b^n c^n \mid n \geq 0 \}$ is not accepted by any* PD-CD-R(1)*-system.*

The language $L_{pal} = \{ wcw^R \mid w \in \{a, b\}^* \}$ is a context-free language that is not a one-counter language (see *e.g.* [2]). As a context-free language it is accepted by some PD-CD-R(1)-system by Proposition 3.4, but based on Corollary 3.9 we can show that it is not accepted by any OC-CD-R(1)-system.

Proposition 3.12. *The language $L_{pal} = \{ wcw^R \mid w \in \{a, b\}^* \}$ is not accepted by any* OC-CD-R(1)*-system.*

Proof. By Corollary 3.9 we only need to show that the language L_{pal} does not contain a sublanguage that is letter-equivalent to L_{pal} itself and that is a one-counter language.

Let $\Sigma = \{a, b, c\}$, and let E be a sublanguage of L_{pal} that is letter-equivalent to L_{pal}. Hence, for all $n \geq 1$,

$$\psi(E \cap \Sigma^{2n+1}) = \{ (2i, 2(n-i), 1) \mid 0 \leq i \leq n \},$$

and accordingly,

$$\psi(E \cap \Sigma^{\leq 2n+1}) = \{ (2i, 2(m-i), 1) \mid 0 \leq m \leq n, 0 \leq i \leq m \}.$$

Assume that there exists a one-counter automaton $M = (Q, \Sigma, \{C\}, q_0, \bot, \delta, F)$ such that $L(M) = E$ holds. Thus, for each $m \in \{1, \ldots, n\}$ and each $i \in \{0, 1, \ldots, m\}$, there exists a word $w(i,m) \in E$ such that

$$\psi(w(i,m)) = (2i, 2(m-i), 1).$$

As E is a sublanguage of L_{pal}, $w(i,m) = u(i,m)cu(i,m)^R$ for some $u(i,m) \in \{a,b\}^m$ satisfying $|u(i,m)|_a = i$ and $|u(i,m)|_b = m - i$. As $E = L(M)$, M has an accepting computation on input $w(i,m)$, which is of the following form:

$$\begin{aligned}(q_0, w(i,m), \bot) &= (q_0, u(i,m)cu(i,m)^R, \bot) \vdash_M^* (q_1, cu(i,m)^R, \bot C^j)\\ &\vdash_M (q_2, u(i,m)^R, \bot C^{j+\mu}) \vdash_M^* (q_f, \varepsilon, \bot)\end{aligned}$$

for some states $q_1, q_2 \in Q$, a final state $q_f \in F$, and integers $j \geq 0$ and $\mu \in \{-1, 0, 1\}$. While processing the prefix $u(i,m)$ of $w(i,m)$, M can increase the value on its counter at most $m = |u(i,m)|$ times, which means that $j \leq m$ holds. Hence, while there are at least

$$\sum_{m=0}^{n} (m+1) = \sum_{m=1}^{n+1} m = \frac{1}{2}(n+1)(n+2)$$

many different words $w = ucu^R$ in E such that $|u| \leq n$, there are only $n+1$ different values that the counter of M may have after processing the prefix u of any of these words. Choose $n > 2 \cdot |Q|$. Then $\frac{1}{2}(n+1)(n+2) > (n+1)|Q|$, which means that there are two different input words $ucu^R, vcv^R \in E$, a state $q_1 \in Q$, and an integer $j \leq n$ such that $|u| \leq n$, $|v| \leq n$, $\psi(u) \neq \psi(v)$, and

$$(q_0, ucu^R, \bot) \vdash_M^* (q_1, cu^R, \bot C^j) \vdash_M^* (q_f, \varepsilon, \bot)$$

and

$$(q_0, vcv^R, \bot) \vdash_M^* (q_1, cv^R, \bot C^j) \vdash_M^* (q'_f, \varepsilon, \bot)$$

are both accepting computations of M. But then also

$$(q_0, ucv^R, \bot) \vdash_M^* (q_1, cv^R, \bot C^j) \vdash_M^* (q'_f, \varepsilon, \bot)$$

is an accepting computation of M. However, $ucv^R \notin L_{pal}$, implying that $ucv^R \notin E$, that is, $L(M) \neq E$. Thus, no sublanguage of L_{pal} can be both, letter-equivalent to L_{pal} and a one-counter language. □

4. Context-free trace languages

Below we only restate those definitions of and basic results on traces, trace monoids and trace languages that we will need for presenting our results. For a detailed presentation of trace theory see the monograph by Diekert and Rozenberg [9].

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

To simplify the notation in what follows, we introduce the following notions. For $w \in \Sigma^*$, we use $\mathrm{Alph}(w)$ to denote the set of all letters that occur in w, that is, $\mathrm{Alph}(w) = \{\, a \in \Sigma \mid |w|_a > 0 \,\}$. Now we extend the independence relation from letters to words by defining, for all words $u, v \in \Sigma^*$,

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

As $\mathrm{Alph}(\varepsilon) = \emptyset$, we see that $(\varepsilon, w) \in I_D$ for every word $w \in \Sigma^*$. The following technical result (see *e.g.* [9] Claim A in the proof of Proposition 6.2.2) will be useful in what follows.

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

A subset S of a trace monoid $M(D)$ is called *recognizable* if there exist a finite monoid N, a morphism $\alpha : M(D) \to N$, and a subset P of N such that $S = \alpha^{-1}(P)$ [3]. Accordingly, this property can be characterized as follows (see [9] Prop. 6.1.10).

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

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

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

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

By $\mathsf{RAT}(M(D))$ we denote the set of rational subsets of $M(D)$. Concerning the relationship between the recognizable subsets of $M(D)$ and the rational subsets of $M(D)$ the following results are known (see *e.g.* [9]).

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

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

For a subset S of the trace monoid $M(D)$ presented by (Σ, D), the set of words $L = \varphi_D^{-1}(S) \subseteq \Sigma^*$ is called the *linearization* of S. A language $L \subseteq \Sigma^*$ is the *linearization* of a rational trace language, if there exists a dependency relation D on Σ such that $L = \varphi_D^{-1}(S)$ for a rational subset S of the trace monoid $M(D)$ presented by (Σ, D). From Proposition 4.3 it follows that L is the linearization of a rational trace language if and only if there exist a trace monoid $M(D)$ and a regular language $R \subseteq \Sigma^*$ such that $L = \varphi_D^{-1}(\varphi_D(R)) = \bigcup_{w \in R}[w]_D$. By $\mathcal{LRAT}(D)$ we denote the set of linearizations of rational trace languages $\{\, \varphi_D^{-1}(S) \mid S \in \mathsf{RAT}(M(D)) \,\}$, and $\mathcal{LRAT}$ is the class of all linearizations of rational trace languages. In [21] the following result was established.

Theorem 4.5. *$\mathcal{LRAT} \subsetneq \mathcal{L}_{=1}(\mathsf{stl\text{-}det\text{-}local\text{-}CD\text{-}R(1)})$, that is, if $M(D)$ is the trace monoid presented by (Σ, D), where D is a dependency relation on the finite alphabet Σ, then the language $\varphi_D^{-1}(S)$ is accepted by a $\mathsf{stl\text{-}det\text{-}local\text{-}CD\text{-}R(1)}$-system working in mode $= 1$ for each rational set S of traces of $M(D)$.*

Here we are interested in more general trace languages. A subset S of the trace monoid $M(D)$ presented by (Σ, D) is called a *one-counter trace language*, if there exists a one-counter language $R \subseteq \Sigma^*$ such that $S = \varphi_D(R)$, and S is called a *context-free trace language*, if there exists a context-free language $R \subseteq \Sigma^*$ such that $S = \varphi_D(R)$ [1, 4]. Then the language $L = \varphi_D^{-1}(S) = \varphi_D^{-1}(\varphi_D(R))$ is the *linearization* of the one-counter or context-free trace language S. By $\mathcal{LOC}(D)$ we denote the set of linearizations of one-counter trace languages obtained from (Σ, D), and $\mathcal{LOC}$ is the class of linearizations of all one-counter trace languages. Further, by $\mathcal{LCF}(D)$ we denote the set of linearizations of context-free trace languages obtained from (Σ, D), and $\mathcal{LCF}$ is the class of linearizations of all context-free trace languages.

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

$$\mathcal{LCF}(D) \subseteq \mathcal{L}(\mathsf{PD\text{-}CD\text{-}R(1)}),$$

that is, the language $\varphi_D^{-1}(\varphi_D(R))$ is accepted by a $\mathsf{PD\text{-}CD\text{-}R(1)}$-system for each context-free language $R \subseteq \Sigma^*$.

Proof. Let D be a dependency relation on Σ, let $R \subseteq \Sigma^*$ be a context-free language, and let $L = \varphi_D^{-1}(\varphi_D(R))$ be the linearization of the context-free trace language $\varphi_D(R)$. Then there exists a context-free grammar $G = (V, \Sigma, S, P)$ in

quadratic Greibach normal form for $R' = R \setminus \{\varepsilon\}$. We can assume without loss of generality that the start symbol S does not occur on the right-hand side of any production.

From G we construct a PD-CD-R(1)-system $\mathcal{M}$ for the language L. This construction is a variant of the construction used in the proof of Proposition 3.4. For each nonterminal $A \in V$ and each terminal $a \in \Sigma$ such that P contains a production of the form $(A \to a\gamma)$, $\mathcal{M}$ has a component automaton $M_{(A,a)}$. As in the proof of Proposition 3.4, the component automata with index A are used to simulate A-productions, but here the additional second index $a \in \Sigma$ is used to differentiate between A-productions based on the terminal symbol being generated: the component automaton $M_{(A,a)}$ only simulates those A-productions for which the right-hand side starts with the terminal a. In addition, $M_{(A,a)}$ reads across all letters $b \in \Sigma$ that are independent of a with respect to the dependency relation D. Thus, $\mathcal{M}$ is essentially a pushdown automaton for the language R that is equipped with the additional ability to skip across independent letters when looking for a particular input symbol a.

The PD-CD-R(1)-system $\mathcal{M} = (I, \Sigma, (M_i, \sigma_i)_{i \in I}, V, \bot, I_0, \delta)$ is defined as follows:

- $I = \{\, (A,a) \mid A \in V, a \in \Sigma, \exists\, \gamma \in V^{\leq 2} : (A \to a\gamma) \in P \,\} \cup \{+\}$,
- $I_0 = \{\, (S,a) \mid \exists\, \gamma \in V^{\leq 2} : (S \to a\gamma) \in P \,\} \cup \{\, + \mid \varepsilon \in R \,\}$,
- the stateless deterministic R(1)-automata $M_{(A,a)}$ $((A,a) \in I)$ and M_+ are defined as follows:

$$
\begin{array}{lll}
(1)\ \delta_{(A,a)}(\text{¢}) = \mathsf{MVR}, & & \\
(2)\ \delta_{(A,a)}(a) = \varepsilon, & & \\
(3)\ \delta_{(A,a)}(b) = \mathsf{MVR} & \text{for all } b \in \Sigma \text{ satisfying } (b,a) \in I_D, & \\
(4)\ \delta_{(A,a)}(b) = \emptyset, & \text{otherwise}, & \\
(5)\ \delta_{(A,a)}(\$) = \emptyset, & & \\
(6)\ \delta_+(\text{¢}) = \mathsf{MVR}, & & \\
(7)\ \delta_+(b) = \emptyset & \text{for all } b \in \Sigma, & \\
(8)\ \delta_+(\$) = \mathsf{Accept}, & &
\end{array}
$$

- the sets of successors are defined as $\sigma_{(A,a)} = \sigma_+ = I$ for all $(A,a) \in I$,
- and the successor relation δ is defined as follows, where $A \in V$ and $a \in \Sigma$:

$$
\begin{aligned}
(1)\ \delta((S,a),a,\bot) = {} & \{\, (+,\bot) \mid (S \to a) \in P \,\} \\
& \cup \{\, ((B,b),\bot B) \mid (S \to aB) \in P, (B,b) \in I \,\} \\
& \cup \{\, ((B,b),\bot CB) \mid (S \to aBC) \in P, (B,b) \in I \,\}, \\
(2)\ \delta((A,a),a,A) = {} & \{\, ((B,b),\varepsilon) \mid B \in V \setminus \{S\}, (B,b) \in I, (A \to a) \in P \,\} \\
& \cup \{\, (+,\varepsilon) \mid (A \to a) \in P \,\} \\
& \cup \{\, ((B,b),B) \mid (A \to aB) \in P, (B,b) \in I \,\} \\
& \cup \{\, ((B,b),CB) \mid (A \to aBC) \in P, (B,b) \in I \,\},
\end{aligned}
$$

and δ yields the empty set for all other values.

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

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

Proof. Assume that $w \in \bigcup_{u \in R}[u]_D$. Then there exists a word $u \in R$ such that $w \equiv_D u$. If $w = \varepsilon$, then $u = \varepsilon$, and therewith $\varepsilon \in R$. From the definition of $\mathcal{M}$ we see that in this case $+ \in I_0$, which implies that $w = \varepsilon \in L(\mathcal{M})$.

So assume that $w \neq \varepsilon$ implying that $u \neq \varepsilon$, either. As $w \equiv_D u$, there exists a sequence of words $u = w_0, w_1, \ldots, w_n = w$ such that, for each $i = 1, \ldots, n$, w_i is obtained from w_{i-1} by replacing a factor ab by ba for some pair of letters $(a, b) \in I_D$. We now prove that $w_i \in L(\mathcal{M})$ for all i by induction on i.

For $i = 0$ we have $w_0 = u \in R$. Then w_0 is generated by the grammar G, and as in the proof of Proposition 3.4, it can be shown that a leftmost derivation $S \Rightarrow_G^+ w_0$ can be simulated by an accepting computation of $\mathcal{M}$ on input w_0. Hence, it follows that w_0 is accepted by $\mathcal{M}$.

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

$$\begin{aligned}((S, a_1), \text{¢}xaby\$, \bot) &\Rightarrow_{\mathcal{M}}^m ((A_1, a), \text{¢}x'aby'\$, \bot\alpha_1)\\ &\Rightarrow_{\mathcal{M}} ((A_2, a_2), \text{¢}x'by'\$, \bot\alpha_2)\\ &\Rightarrow_{\mathcal{M}}^* (+, \text{¢}\$, \bot)\\ &\Rightarrow_{\mathcal{M}} (+, \mathsf{Accept}, \bot)\end{aligned}$$

or

$$\begin{aligned}((S, b_1), \text{¢}xaby\$, \bot) &\Rightarrow_{\mathcal{M}}^m ((A_1, b), \text{¢}x'aby'\$, \bot\beta_1)\\ &\Rightarrow_{\mathcal{M}} ((B_2, b_2), \text{¢}x'ay'\$, \bot\beta_2)\\ &\Rightarrow_{\mathcal{M}}^* (+, \text{¢}\$, \bot)\\ &\Rightarrow_{\mathcal{M}} (+, \mathsf{Accept}, \bot),\end{aligned}$$

where in the first m cycles some letters from x and y are deleted, in this way reducing these factors to x' and y', respectively. However, as $(a, b) \in I$, the component automaton $M_{(A_1,a)}$ can read across the letter b when looking for the leftmost occurrence of the letter a. Thus, $\mathcal{M}$ also has an accepting computation for $w_{i+1} = xbay$, which is of one of the following two forms:

$$\begin{aligned}((S, a_1), \text{¢}xbay\$, \bot) &\Rightarrow_{\mathcal{M}}^m ((A_1, a), \text{¢}x'bay'\$, \bot\alpha_1)\\ &\Rightarrow_{\mathcal{M}} ((A_2, a_2), \text{¢}x'by'\$, \bot\alpha_2)\\ &\Rightarrow_{\mathcal{M}}^* (+, \text{¢}\$, \bot)\\ &\Rightarrow_{\mathcal{M}} (+, \mathsf{Accept}, \bot)\end{aligned}$$

or

$$\begin{aligned}((S, b_1), \text{¢}xbay\$, \bot) &\Rightarrow_{\mathcal{M}}^m ((A_1, b), \text{¢}x'bay'\$, \bot\beta_1)\\ &\Rightarrow_{\mathcal{M}} ((B_2, b_2), \text{¢}x'ay'\$, \bot\beta_2)\\ &\Rightarrow_{\mathcal{M}}^* (+, \text{¢}\$, \bot)\\ &\Rightarrow_{\mathcal{M}} (+, \mathsf{Accept}, \bot),\end{aligned}$$

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

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

Proof. From the definition of $\mathcal{M}$ we see that $\varepsilon \in L(\mathcal{M})$ holds if and only if $\varepsilon \in R$. So let $\varepsilon \neq w \in L(\mathcal{M})$, and let

$$\begin{aligned}((S, a_n), \mathbb{¢}w\$, \bot) &\Rightarrow_{\mathcal{M}} ((A_{n-1}, a_{n-1}), \mathbb{¢}z_{n-1}\$, \bot\alpha_{n-1})\\ &\Rightarrow_{\mathcal{M}} ((A_{n-2}, a_{n-2}), \mathbb{¢}z_{n-2}\$, \bot\alpha_{n-2})\\ &\Rightarrow^*_{\mathcal{M}} \quad ((A_2, a_2), \mathbb{¢}z_2\$, \bot\alpha_2)\\ &\Rightarrow_{\mathcal{M}} \quad ((A_1, a_1), \mathbb{¢}z_1\$, \bot\alpha_1)\\ &\Rightarrow_{\mathcal{M}} \quad (+, \mathbb{¢}\$, \bot)\\ &\Rightarrow_{\mathcal{M}} \quad (+, \mathsf{Accept}, \bot)\end{aligned}$$

be an accepting computation of $\mathcal{M}$ on input $w = z_n$. If $n = 1$, then $A_1 = S$, and we see from the definition of δ that $(+, \bot) \in \delta((S, a_1), a_1, \bot)$ implies that $(S \to a_1) \in P$, that is, $w = z_1 = a_1 \in R$.

For $n > 1$, we claim that, for each $i = 1, \ldots, n-1$, there exists a word $u_i \in \Sigma^*$ such that $u_i \equiv_D z_i$ and $\alpha_i^R \Rightarrow_G^* u_i$. We prove this claim by induction on i.

For $i = 1$ we have $z_i = a_1$. As $n > 1$, $A_1 \neq S$. From the reduction step

$$((A_1, a_1), \mathbb{¢}z_1\$, \bot\alpha_1) \Rightarrow_{\mathcal{M}} (+, \mathbb{¢}\$, \bot),$$

we see that $\alpha_1 = A_1$ and $(+, \varepsilon) \in \delta((A_1, a_1), a_1, A_1)$. Hence, $(A_1 \to a_1) \in P$, and hence, we have $\alpha_1^R = A_1 \Rightarrow_G a_1 = u_1 = z_1$.

Now assume that, for some $i \in \{1, \ldots, n-2\}$, we have a word $u_i \in \Sigma^+$ such that $u_i \equiv_D z_i$ and $\alpha_i^R \Rightarrow_G^* u_i$ hold. The above computation of $\mathcal{M}$ contains the step

$$((A_{i+1}, a_{i+1}), \mathbb{¢}z_{i+1}\$, \bot\alpha_{i+1}) \Rightarrow_{\mathcal{M}} ((A_i, a_i), \mathbb{¢}z_i\$, \bot\alpha_i).$$

Thus, $z_{i+1} = ua_{i+1}v$ and $z_i = uv$ for some $u \in \Sigma_{\mathrm{M}}^{(A_{i+1}, a_{i+1})^*}$ and $v \in \Sigma^*$, and $\alpha_{i+1} = \beta A_{i+1}$ and $\alpha_i = \beta\eta$, where $((A_i, a_i), \eta) \in \delta((A_{i+1}, a_{i+1}), a_{i+1}, A_{i+1})$. Hence, $(A_{i+1} \to a_{i+1}\eta^R) \in P$, which implies that

$$\alpha_{i+1}^R = A_{i+1}\beta^R \Rightarrow_G a_{i+1}\eta^R\beta^R = a_{i+1}\alpha_i^R \Rightarrow_G^* a_{i+1}u_i.$$

As u_i is letter-equivalent to $z_i = uv$, we see that $u_{i+1} = a_{i+1}u_i$ is letter-equivalent to $z_{i+1} = ua_{i+1}v$. Further, as $u \in \Sigma_{\mathrm{M}}^{(A_{i+1}, a_{i+1})^*}$, $(b, a_{i+1}) \in I_D$ for all letters b occurring in u, which means that

$$u_{i+1} = a_{i+1}u_i \equiv_D a_{i+1}z_i = a_{i+1}uv \equiv_D ua_{i+1}v = z_{i+1}.$$

This completes the inductive step.

Finally, from $((S, a_n), \mathbb{¢}w\$, \bot) \Rightarrow_{\mathcal{M}} ((A_{n-1}, a_{n-1}), \mathbb{¢}z_{n-1}\$, \bot\alpha_{n-1})$, we see that $w = u'a_nv'$ and $z_{n-1} = u'v'$ for some word u' satisfying $(u', a_n) \in I_D$, and $((A_{n-1}, a_{n-1}), \bot\alpha_{n-1}) \in \delta((S, a_n), a_n, \bot)$. Hence, $(S \to a_n\alpha_{n-1}^R) \in P$, and we obtain

$$S \Rightarrow_G a_n\alpha_{n-1}^R \Rightarrow_G^* a_nu_{n-1} = u_n,$$

that is, $u_n \in R$. Further, $u_n = a_nu_{n-1} \equiv_D a_nz_{n-1} = a_nu'v' \equiv_D u'a_nv' = w$. This completes the proof of Claim 2. □

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

An analogous result also holds for one-counter languages.

Theorem 4.7. *Let $M(D)$ be the trace monoid presented by (Σ, D), where D is a dependency relation on the finite alphabet Σ. Then*

$$\mathcal{LOC}(D) \subseteq \mathcal{L}(\mathsf{OC\text{-}CD\text{-}R(1)}),$$

that is, the language $\varphi_D^{-1}(\varphi_D(R))$ is accepted by an OC-CD-R(1)*-system for each one-counter language $R \subseteq \Sigma^*$.*

Proof. Let $R \subseteq \Sigma^*$ be a one-counter language, and let $L = \varphi_D^{-1}(\varphi_D(R))$. Then there exists a one-counter automaton $\mathcal{A} = (Q, \Sigma, \{C\}, q_0, \bot, \delta_{\mathcal{A}}, F)$ for R, that is, for all $w \in \Sigma^*$, $w \in R$ if and only if $(q_0, w, \bot) \vdash_{\mathcal{A}}^* (q, \varepsilon, \bot)$ holds for some final state $q \in F$.

From $\mathcal{A}$ we construct an OC-CD-R(1)-system $\mathcal{M}$ for L. This construction is a variant of the construction used in the proof of Proposition 3.5. For each state $q \in Q$ and each terminal $a \in \Sigma$, $\mathcal{M}$ has two component automata: $M_{(q,=,a)}$ and $M_{(q,>,a)}$. As in the proof of Proposition 3.5, the component automata with index $(q, =)$ are used to simulate the transitions that $\mathcal{A}$ executes in state q with empty counter, and the component automata with index $(q, >)$ are used to simulate the transitions that $\mathcal{A}$ executes in state q with non-empty counter. Here, however, the additional third index $a \in \Sigma$ is used to differentiate between the transitions of $\mathcal{A}$ based on the symbol read: the component automata $M_{(q,=,a)}$ and $M_{(q,>,a)}$ only simulate those transitions of $\mathcal{A}$ that read the symbol a. In addition, these component automata read across all letters $b \in \Sigma$ that are independent of a with respect to the dependency relation D. Thus, $\mathcal{M}$ is essentially a one-counter automaton for the language R that is equipped with the additional ability to skip across independent letters when looking for a particular input symbol a.

The system $\mathcal{M} = (I, \Sigma, (M_i, \sigma_i)_{i \in I}, \{C\}, \bot, I_0, \delta)$ is defined as follows:

- $I = (Q \times \{=, >\} \times \Sigma) \cup \{+\}$, $I_0 = \{ (q_0, =, a) \mid a \in \Sigma \} \cup \{ + \mid \varepsilon \in R \}$,
- $\sigma_{(q,>,a)} = \sigma_{(q,=,a)} = \sigma_+ = I$ for all $q \in Q$ and all $a \in \Sigma$,
- the stateless deterministic R(1)-automata $M_{(q,>,a)}$ and $M_{(q,=,a)}$ $(q \in Q, a \in \Sigma)$ are defined as follows:

$$\begin{aligned}
&(1)\ \delta_{(q,=,a)}(¢) = \mathsf{MVR},\\
&(2)\ \delta_{(q,=,a)}(a) = \varepsilon, \quad \text{if } \delta_{\mathcal{A}}(q, a, \bot) \text{ is defined},\\
&(3)\ \delta_{(q,=,a)}(b) = \mathsf{MVR} \quad \text{for all } b \in \Sigma \text{ satisfying } (b, a) \in I_D,\\
&(4)\ \delta_{(q,=,a)}(b) = \emptyset, \quad \text{otherwise},\\
&(5)\ \delta_{(q,=,a)}(\$) = \emptyset,\\
&(6)\ \delta_{(q,>,a)}(¢) = \mathsf{MVR},\\
&(7)\ \delta_{(q,>,a)}(a) = \varepsilon, \quad \text{if } \delta_{\mathcal{A}}(q, a, C) \text{ is defined},\\
&(8)\ \delta_{(q,>,a)}(b) = \mathsf{MVR} \quad \text{for all } b \in \Sigma \text{ satisfying } (b, a) \in I_D,\\
&(9)\ \delta_{(q,>,a)}(b) = \emptyset, \quad \text{otherwise},\\
&(10)\ \delta_{(q,>,a)}(\$) = \emptyset,
\end{aligned}$$

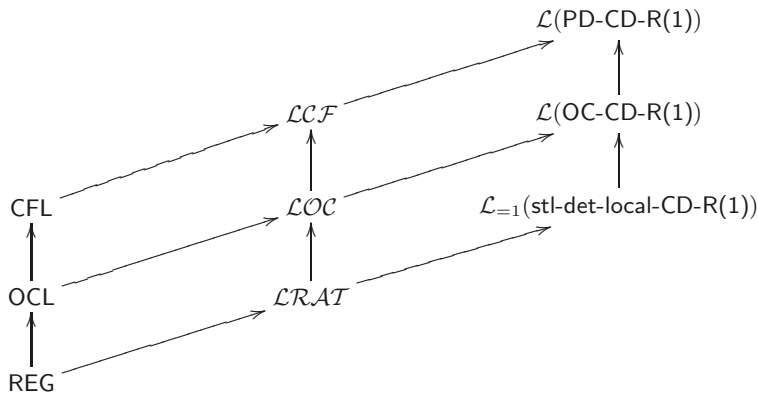

FIGURE 1. Hierarchy of language classes accepted by various types of CD-R(1)-systems. Each arrow represents a proper inclusion, and classes that are not connected by a sequence of arrows are incomparable under inclusion.

- M_+ is defined by

$$\delta_+(¢) = \mathsf{MVR},\ \delta_+(b) = \emptyset \text{ for all } b \in \Sigma, \text{ and } \delta_+(\$) = \mathsf{Accept},$$

- and δ is defined as follows, where $q \in Q$, $a, b \in \Sigma$, and $i \in \{1, 2\}$:

$$\begin{aligned}
(1)\ \delta((q,=,a),a,\bot) =& \{\, ((q',=,b),\bot) \mid (q',\bot) \in \delta_{\mathcal{A}}(q,a,\bot),\ b \in \Sigma \,\} \\
&\cup \{\, (+,\bot) \mid \exists q' \in F : (q',\bot) \in \delta_{\mathcal{A}}(q,a,\bot) \,\} \\
&\cup \{\, ((q',>,b),\bot C) \mid (q',\bot C) \in \delta_{\mathcal{A}}(q,a,\bot),\ b \in \Sigma \,\}, \\
(2)\ \delta((q,>,a),a,C) =& \{\, ((q',>,b),C^i) \mid (q',C^i) \in \delta_{\mathcal{A}}(q,a,C),\ b \in \Sigma \,\} \\
&\cup \{\, ((q',>,b),\varepsilon) \mid (q',\varepsilon) \in \delta_{\mathcal{A}}(q,a,C),\ b \in \Sigma \,\} \\
&\cup \{\, ((q',=,b),\varepsilon) \mid (q',\varepsilon) \in \delta_{\mathcal{A}}(q,a,C),\ b \in \Sigma \,\} \\
&\cup \{\, (+,\varepsilon) \mid \exists q' \in F : (q',\varepsilon) \in \delta_{\mathcal{A}}(q,a,C) \,\},
\end{aligned}$$

while δ yields the empty set for all other values. As in the proof of Theorem 4.6 it can now be shown that $L(\mathcal{M}) = \varphi_D^{-1}(\varphi_D(R)) = \bigcup_{u \in R}[u]_D$ holds. □

Thus, we have the inclusions of language classes depicted in the diagram in Figure 1. As $\mathcal{L}_{=1}$(stl-det-local-CD-R(1)) contains the non-context-free language $\{\, w \in \{a,b,c\}^* \mid |w|_a = |w|_b = |w|_c \geq 0 \,\}$, and as this class does not contain the one-counter language $\{\, a^n b^n \mid n \geq 0 \,\}$, we see that $\mathcal{L}_{=1}$(stl-det-local-CD-R(1)) is incomparable under inclusion to the language classes OCL and CFL. From Proposition 3.12 we see that the class $\mathcal{L}$(OC-CD-R(1)) is incomparable under inclusion to the language class CFL.

Let $\Sigma = \{a,b,c\}$, and let $L' = \{\, wa^m \mid |w|_a = |w|_b = |w|_c \geq 1, m \geq 1 \,\}$. As shown in Example 4 of [22] the language L' is accepted by a stl-det-local-CD-R(1)-system. However, L' is not the linearization of a context-free trace language.

Proposition 4.8. *For each dependency relation D on Σ and each context-free language $R \subseteq \Sigma^*$, $L' \neq \bigcup_{w \in R}[w]_D$.*

Proof. Let D be a dependency relation on Σ, and let $R \subseteq \Sigma^*$ be a language such that $L' = \bigcup_{w \in R}[w]_D$ holds. We claim that from these assumptions it follows that R is not context-free.

Claim 1. $(a,b),(b,a) \in D$.

Proof. Assume that $(a,b) \notin D$. As D is symmetric, this means that $(b,a) \notin D$, either. Hence, $(a,b),(b,a) \in I_D$ implying that $ab \equiv_D ba$ holds. For all $n,m \geq 1$, the word $c^n(ab)^n a^m \in L'$, and hence, there exists a word $u(n,m) \in R$ such that

$$u(n,m) \equiv_D c^n(ab)^n a^m \equiv_D c^n a^{n+m} b^n.$$

This, however, contradicts the assumption $L' = \bigcup_{w \in R}[w]_D$, as $c^n a^{n+m} b^n \notin L'$. Thus, it follows that $(a,b),(b,a) \in D$. □

Claim 2. $(a,c),(c,a) \in D$.

Proof. Assume that $(a,c) \notin D$. As D is symmetric, this means that $(c,a) \notin D$, either. Hence, $(a,c),(c,a) \in I_D$ implying that $ac \equiv_D ca$ holds. For all $n,m \geq 1$, the word $b^n(ac)^n a^m \in L'$, and hence, there exists a word $v(n,m) \in R$ such that

$$v(n,m) \equiv_D b^n(ac)^n a^m \equiv_D b^n a^{n+m} c^n.$$

This, however, contradicts the assumption $L' = \bigcup_{w \in R}[w]_D$, as $b^n a^{n+m} c^n \notin L'$. Thus, it follows that $(a,c),(c,a) \in D$. □

Accordingly, we see that

$$D = \{(a,a),(b,b),(c,c),(a,b),(b,a),(a,c),(c,a)\}$$

or

$$D = \{(a,a),(b,b),(c,c),(a,b),(b,a),(a,c),(c,a),(b,c),(c,b)\}.$$

For all $n,m \geq 1$, the word $b^n a^n c^n a^m \in L'$, and hence, there exists a word $w(n,m) \in R$ such that $w(n,m) \equiv_D b^n a^n c^n a^m$. However, in each of these two cases we see that $[b^n a^n c^n a^m]_D = \{b^n a^n c^n a^m\}$, as $(b,a),(a,c),(c,a) \in D$, that is, $w(n,m) = b^n a^n c^n a^m$. Thus,

$$R \cap (b^+ \cdot a^+ \cdot c^+ \cdot a^+) = \{\, b^n a^n c^n a^m \mid n,m \geq 1 \,\},$$

which is not context-free. As the class of context-free languages is closed under the operation of intersection with a regular language, it follows that R is not context-free. This completes the proof of Proposition 4.8. □

Together with Proposition 3.12 this result has the following consequences.

Corollary 4.9.
(a) $\mathcal{LCF}$ *is incomparable under inclusion to* $\mathcal{L}_{=1}(\mathsf{stl\text{-}det\text{-}local\text{-}CD\text{-}R(1)})$.
(b) $\mathcal{LCF}$ *is incomparable under inclusion to* $\mathcal{L}(\mathsf{OC\text{-}CD\text{-}R(1)})$.
(c) $\mathcal{LOC}$ *is incomparable under inclusion to* $\mathcal{L}_{=1}(\mathsf{stl\text{-}det\text{-}local\text{-}CD\text{-}R(1)})$.
(d) $\mathcal{LCF} \subsetneq \mathcal{L}(\mathsf{PD\text{-}CD\text{-}R(1)})$.
(e) $\mathcal{LOC} \subsetneq \mathcal{L}(\mathsf{OC\text{-}CD\text{-}R(1)})$.

Next we present a restricted class of PD-CD-R(1)-systems that accept exactly the linearizations of context-free trace languages.

Definition 4.10. Let $\mathcal{M} = (I, \Sigma, (M_i, \sigma_i)_{i\in I}, \Gamma, \bot, I_0, \delta)$ be a PD-CD-R(1)-system in strong normal form that satisfies the following condition:

$$(*) \qquad \forall i, j \in I : \Sigma_\varepsilon^{(i)} = \Sigma_\varepsilon^{(j)} \text{ implies that } \Sigma_{\mathrm{M}}^{(i)} = \Sigma_{\mathrm{M}}^{(j)},$$

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

$$I_{\mathcal{M}} = \bigcup_{i\in I} (\Sigma_{\mathrm{M}}^{(i)} \times \Sigma_\varepsilon^{(i)}),$$

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

Observe that the relation $I_{\mathcal{M}}$ defined above is necessarily irreflexive, but that it will in general not be symmetric.

Theorem 4.11. *Let* $\mathcal{M}$ *be a* PD-CD-R(1)*-system over* Σ *satisfying condition* $(*)$ *above. If the associated relation* $I_{\mathcal{M}}$ *is symmetric, then* $L(\mathcal{M}) \in \mathcal{LCF}(D_M)$*, that is,* $L(\mathcal{M})$ *is the linearization of a context-free trace language. In fact, from* $\mathcal{M}$ *one can construct a pushdown automaton* B *over* Σ *such that* $L(\mathcal{M}) = \bigcup_{u\in L(B)} [u]_{D_{\mathcal{M}}}$.

Proof. Let $\mathcal{M} = (I, \Sigma, (M_i, \sigma_i)_{i\in I}, \Gamma, \bot, I_0, \delta)$ be a PD-CD-R(1)-system in strong normal form. Assume that $\mathcal{M}$ satisfies condition $(*)$ above, and that the associated relation $I_{\mathcal{M}} = \bigcup_{i\in I} (\Sigma_{\mathrm{M}}^{(i)} \times \Sigma_\varepsilon^{(i)})$ is symmetric. Then the relation $D_{\mathcal{M}} = (\Sigma \times \Sigma) \setminus I_{\mathcal{M}}$ is reflexive and symmetric, and so it is a dependency relation on Σ with associated independence relation $I_{\mathcal{M}}$. Without loss of generality we may assume that all letters from Σ do actually occur in some words of $L(\mathcal{M})$, since otherwise we could simply remove these letters from Σ. From the properties of $\mathcal{M}$ we obtain the following consequences:

(1) As all words $w \in L(\mathcal{M})$ are first reduced to the empty word, which is then accepted by the accepting component automaton M_+ of $\mathcal{M}$, we see that, for each letter $a \in \Sigma$, there exists a component automaton M_i such that $\Sigma_\varepsilon^{(i)} = \{a\}$.

(2) If $(a,b) \in I_{\mathcal{M}}$, then $a \in \Sigma_{\mathrm{M}}^{(i)}$ for all component automata M_i for which $\Sigma_{\varepsilon}^{(i)} = \{b\}$ holds.
(3) If $(a,b) \in I_{\mathcal{M}}$, then $(b,a) \in I_{\mathcal{M}}$, too, and hence, $b \in \Sigma_{\mathrm{M}}^{(j)}$ for all component automata M_j for which $\Sigma_{\varepsilon}^{(j)} = \{a\}$ holds.

Let $L = L(\mathcal{M})$. We claim that L is the linearization of a context-free trace language over the trace monoid defined by $(\Sigma, D_{\mathcal{M}})$. To verify this claim we present a context-free language $R \subseteq \Sigma^*$ such that $L = \bigcup_{u \in R} [u]_{D_{\mathcal{M}}}$. The context-free language R will be defined through a nondeterministic pushdown automaton $B = (Q, \Sigma, \Gamma, q_0, \bot, \delta_B, F)$ which is obtained as follows:

- $Q = I \cup \{q_0\}$, where q_0 is a new state,
- $F = \{+\}$, which corresponds to the unique accepting component M_+ of $\mathcal{M}$, and
- the transition relation δ_B is defined as follows for all $i \in I_r = I \setminus \{+\}$, $a \in \Sigma$, and $A \in \Gamma$:

$$\begin{aligned}
&(1)\ \delta_B(q_0, \varepsilon, \bot) = \{\, (i, \bot) \mid i \in I_0 \,\},\\
&(2)\ \delta_B(i, a, \bot) \ = \{\, (j, \bot\eta) \mid (j, \bot\eta) \in \delta(i, a, \bot) \,\},\\
&(3)\ \delta_B(i, a, A) \ = \{\, (j, \alpha) \mid (j, \alpha) \in \delta(i, a, A) \,\},
\end{aligned}$$

and $\delta_B(+, a, A)$ is undefined for all $a \in \Sigma$ and all $A \in \Gamma \cup \{\bot\}$.

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

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

Proof. The above construction is identical to the one used in the proof of Theorem 3.8. Thus, it follows that $R = L(B)$ is a sublanguage of $L = L(\mathcal{M})$ that is letter-equivalent to L.

Now assume that $w = xaby \in L$ and that $z = xbay$ for a pair of letters $(a,b) \in I_{\mathcal{M}}$. Then $\mathcal{M}$ has an accepting computation for $w = xaby$, which is of one of the following two forms:

$$\begin{aligned}
(i_0, \mathbb{c}xaby\$, \bot) &\Rightarrow_{\mathcal{M}}^{k} (i_1, \mathbb{c}x_1aby_1\$, \bot\alpha_1) \Rightarrow_{\mathcal{M}} (i_2, \mathbb{c}x_1by_1\$, \bot\alpha_2)\\
&\Rightarrow_{\mathcal{M}}^{l} (i_3, \mathbb{c}x_2by_2\$, \bot\alpha_3) \Rightarrow_{\mathcal{M}} (i_4, \mathbb{c}x_2y_2\$, \bot\alpha_4)\\
&\Rightarrow_{\mathcal{M}}^{*} (+, \mathbb{c}\$, \bot) \Rightarrow_{\mathcal{M}} (+, \mathsf{Accept}, \bot),
\end{aligned}$$

or

$$\begin{aligned}
(i_0, \mathbb{c}xaby\$, \bot) &\Rightarrow_{\mathcal{M}}^{k} (j_1, \mathbb{c}x_1aby_1\$, \bot\alpha_1) \Rightarrow_{\mathcal{M}} (j_2, \mathbb{c}x_1ay_1\$, \bot\alpha_2)\\
&\Rightarrow_{\mathcal{M}}^{l} (j_3, \mathbb{c}x_2ay_2\$, \bot\alpha_3) \Rightarrow_{\mathcal{M}} (j_4, \mathbb{c}x_2y_2\$, \bot\alpha_4)\\
&\Rightarrow_{\mathcal{M}}^{*} (+, \mathbb{c}\$, \bot) \Rightarrow_{\mathcal{M}} (+, \mathsf{Accept}, \bot),
\end{aligned}$$

where in the first k cycles some letters from x and y are deleted, in this way reducing these factors to x_1 and y_1, respectively, $\Sigma_{\varepsilon}^{(i_1)} = \{a\} = \Sigma_{\varepsilon}^{(j_3)}$ and $\Sigma_{\varepsilon}^{(i_3)} = \{b\} = \Sigma_{\varepsilon}^{(j_1)}$, and in the latter l cycles some letters from x_1 and y_1 are deleted, reducing these factors to x_2 and y_2, respectively. As $(a,b) \in I_{\mathcal{M}}$, we see from the

above stated properties of $\mathcal{M}$ that $b \in \Sigma_{\mathrm{M}}^{(i_1)}$. Hence, in the former case we obtain the computation

$$\begin{aligned}(i_0, \mathbb{¢}xbay\$, \bot) &\Rightarrow_{\mathcal{M}}^{k} (i_1, \mathbb{¢}x_1bay_1\$, \bot\alpha_1) \Rightarrow_{\mathcal{M}} (i_2, \mathbb{¢}x_1by_1\$, \bot\alpha_2)\\ &\Rightarrow_{\mathcal{M}}^{l} (i_3, \mathbb{¢}x_2by_2\$, \bot\alpha_3) \Rightarrow_{\mathcal{M}} (i_4, \mathbb{¢}x_2y_2\$, \bot\alpha_4)\\ &\Rightarrow_{\mathcal{M}}^{*} (+, \mathbb{¢}\$, \bot) \Rightarrow_{\mathcal{M}} (+, \mathsf{Accept}, \bot),\end{aligned}$$

while in the latter case we have the computation

$$\begin{aligned}(i_0, \mathbb{¢}xbay\$, \bot) &\Rightarrow_{\mathcal{M}}^{k} (j_1, \mathbb{¢}x_1bay_1\$, \bot\alpha_1) \Rightarrow_{\mathcal{M}} (j_2, \mathbb{¢}x_1ay_1\$, \bot\alpha_2)\\ &\Rightarrow_{\mathcal{M}}^{l} (j_3, \mathbb{¢}x_2ay_2\$, \bot\alpha_3) \Rightarrow_{\mathcal{M}} (j_4, \mathbb{¢}x_2y_2\$, \bot\alpha_4)\\ &\Rightarrow_{\mathcal{M}}^{*} (+, \mathbb{¢}\$, \bot) \Rightarrow_{\mathcal{M}} (+, \mathsf{Accept}, \bot).\end{aligned}$$

Hence, it follows that $z \in L$.

Finally, if $w \equiv_{D_{\mathcal{M}}} u$ holds for some $u \in L$, then there exists a sequence $u = w_0, w_1, \ldots, w_n = w$ such that, for each $i = 1, \ldots, n$, w_i is obtained from w_{i-1} by replacing a factor ab by ba for some pair of letters $(a, b) \in I_{\mathcal{M}}$. Hence, it follows from the observation above that $w_i \in L$ for all $i = 1, \ldots, n$, which in turn shows that $\bigcup_{u \in R}[u]_{D_{\mathcal{M}}} \subseteq L$ holds. □

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

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

$$\begin{aligned}(i_n, \mathbb{¢}w_n\$, \bot) &\Rightarrow_{\mathcal{M}} (i_{n-1}, \mathbb{¢}w_{n-1}\$, \bot\alpha_{n-1}) \Rightarrow_{\mathcal{M}} \quad \ldots\\ &\Rightarrow_{\mathcal{M}} (i_1, \mathbb{¢}w_1\$, \bot\alpha_1) \Rightarrow_{\mathcal{M}} (+, \mathbb{¢}\$, \bot)\\ &\Rightarrow_{\mathcal{M}} (+, \mathsf{Accept}, \bot)\end{aligned}$$

be an accepting computation of $\mathcal{M}$ on input $w = w_n$. We claim that, for each $j = 1, \ldots, n$, there exists a word $u_j \in \Sigma^*$ such that $u_j \equiv_{D_{\mathcal{M}}} w_j$, and the pushdown automaton B accepts when starting from the configuration $(i_j, u_j, \bot\alpha_j)$.

We prove this claim by induction on j. For $j = 1$ we have $w_j = a_1 \in \Sigma$, where $\Sigma_{\varepsilon}^{(i_1)} = \{a_1\}$, and either $\alpha_1 = \varepsilon$ and $(+, \bot) \in \delta(i_1, a_1, \bot)$, or $\alpha_1 = A$ for some $A \in \Gamma$ and $(+, \varepsilon) \in \delta(i_1, a_1, A)$. From the definition of B we see that in either case $(i_1, w_1, \bot\alpha_1) = (i_1, a_1, \bot\alpha_1) \vdash_B (+, \varepsilon, \bot)$ holds. Hence, we simply take $u_1 = a_1 = w_1$, and then the above is an accepting computation of B starting from the configuration $(i_1, u_1, \bot\alpha_1)$.

Now assume that, for some $j \geq 1$, $u_j \equiv_{D_{\mathcal{M}}} w_j$, and that B accepts when starting from the configuration $(i_j, u_j, \bot\alpha_j)$. The above computation of $\mathcal{M}$ contains the step $(i_{j+1}, \mathbb{¢}w_{j+1}\$, \bot\alpha_{j+1}) \Rightarrow_{\mathcal{M}} (i_j, \mathbb{¢}w_j\$, \bot\alpha_j)$. Thus, $w_{j+1} = xa_{j+1}y$ and $w_j = xy$ for some words $x, y \in \Sigma^*$ and a letter a_{j+1} satisfying $\Sigma_{\varepsilon}^{(i_{j+1})} = \{a_{j+1}\}$, and $\alpha_{j+1} = \varepsilon$ and $\alpha_j = \eta$ such that $(i_j, \bot\eta) \in \delta(i_{j+1}, a_{j+1}, \bot)$, or $\alpha_{j+1} = \gamma A$ for some $\gamma \in \Gamma^*$ and some $A \in \Gamma$ and $\alpha_j = \gamma\eta$ such that $(i_j, \eta) \in \delta(i_{j+1}, a_{j+1}, A)$. Also we see that $(x, a_{j+1}) \in I_{\mathcal{M}}$. Again from the definition of B it follows that in either

case B can perform the computational step $(i_{j+1}, a_{j+1}u_j, \bot\alpha_{j+1}) \vdash_B (i_j, u_j, \bot\alpha_j)$. Now let u_{j+1} be the word $u_{j+1} = a_{j+1}u_j$. Then

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

and B has an accepting computation starting from the configuration $(i_{j+1}, u_{j+1}, \bot\alpha_{j+1})$.

Finally, for $j = n$ we obtain a word u such that $u \equiv_{D_\mathcal{M}} w$ and B has an accepting computation starting from the configuration $(i_n, u, \bot)$. As $i_n \in I_0$, this means that $u \in R = L(B)$, as $(q_0, u, \bot) \vdash_B (i_n, u, \bot)$. □

Now Claims 1 and 2 together show that $L = L_{=1}(\mathcal{M}) = \bigcup_{u\in R}[u]_{D_\mathcal{M}}$, which completes the proof of Theorem 4.11. □

If the given PD-CD-R(1)-system $\mathcal{M}$ is a OC-CD-R(1)-system, then the pushdown automaton B constructed in the proof above can easily be turned into a one-counter automaton by deleting the transition $\delta_B(q_0, \varepsilon, \bot) = \{ (i, \bot) \mid i \in I_0 \}$ and by defining $\delta_B(q_0, a, \bot) = \{ (j, \bot\eta) \mid \exists i \in I_0 : (j, \bot\eta) \in \delta(i, a, \bot) \}$ for all $a \in \Sigma$. Thus, we also have the following result.

Corollary 4.12. *Let $\mathcal{M}$ be a* OC-CD-R(1)*-system over Σ satisfying condition $(*)$ above. If the associated relation $I_\mathcal{M}$ is symmetric, then $L(\mathcal{M}) \in \mathcal{LOC}(D_M)$, that is, $L(\mathcal{M})$ is the linearization of a one-counter trace language. In fact, from $\mathcal{M}$ one can construct a one-counter automaton B over Σ such that $L(\mathcal{M}) = \bigcup_{u\in L(B)}[u]_{D_\mathcal{M}}$.*

Observe that the system $\mathcal{M}$ constructed in the proof of Theorem 4.6 is in strong normal form, that it satisfies property $(*)$, and that the associated relation $I_\mathcal{M}$ coincides with the relation I_D, and hence, it is symmetric. Thus, Theorems 4.6, 4.7 and 4.11 together with Corollary 4.12 yield the following characterizations.

Corollary 4.13.

(a) *A language $L \subseteq \Sigma^*$ is the linearization of a one-counter trace language if and only if there exists a* OC-CD-R(1)*-system $\mathcal{M}$ in strong normal form satisfying condition $(*)$ such that the relation $I_\mathcal{M}$ is symmetric and $L = L(\mathcal{M})$.*

(b) *A language $L \subseteq \Sigma^*$ is the linearization of a context-free trace language if and only if there exists a* PD-CD-R(1)*-system $\mathcal{M}$ in strong normal form satisfying condition $(*)$ such that the relation $I_\mathcal{M}$ is symmetric and $L = L(\mathcal{M})$.*

Notice, however, that a PD-CD-R(1)-system may accept the linearization of a context-free trace language, even if it does not satisfy all of the additional restrictions in the characterization above. This observation raises the following questions that currently remain open: Is it decidable whether the language $L(\mathcal{M})$ of a given PD-CD-R(1)-system $\mathcal{M}$ is the linearization of a context-free trace language? Is there possibly a syntactical characterization for those PD-CD-R(1)-systems that accept linearizations of context-free trace languages?

5. Closure and non-closure properties

As seen in Example 3.2 the language

$$L = \{\, a^n v \mid v \in \{b,c\}^*, |v|_b = |v|_c = n, n \geq 0 \,\}$$

is accepted by an OC-CD-R(1)-system, while the language

$$L \cap a^* \cdot b^* \cdot c^* = \{\, a^n b^n c^n \mid n \geq 0 \,\}$$

is not accepted by any PD-CD-R(1)-system (Prop. 3.11). This gives the following non-closure result.

Corollary 5.1. *The language classes $\mathcal{L}$(OC-CD-R(1)) and $\mathcal{L}$(PD-CD-R(1)) are not closed under intersection with regular languages.*

Next we consider the closure under Boolean operations.

Proposition 5.2.

(a) *The language classes $\mathcal{L}$(OC-CD-R(1)) and $\mathcal{L}$(PD-CD-R(1)) are closed under union.*

(b) *The language classes $\mathcal{L}$(OC-CD-R(1)) and $\mathcal{L}$(PD-CD-R(1)) are neither closed under intersection nor under complementation.*

Proof. It is easily seen that these language classes are closed under union, as a PD-CD-R(1)-system for the union $L_1 \cup L_2$ of L_1 and L_2 can immediately be constructed from PD-CD-R(1)-systems for L_1 and L_2. On the other hand, each regular language is accepted by an OC-CD-R(1)-system. Hence, Corollary 5.1 shows that these language classes are not closed under intersection. Finally, closure under union and non-closure under intersection imply that these classes are not closed under complementation, either. □

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

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

If L is accepted by a PD-CD-R(1)-system $\mathcal{M}$, then from $\mathcal{M}$ we can construct a pushdown automaton B for a context-free sublanguage E of L that is letter-equivalent to L (Thm. 3.8). Obviously, the commutative closure $\mathrm{com}(L)$ of L coincides with the commutative closure $\mathrm{com}(E)$ of E. For the dependency relation $D = \{\, (a,a) \mid a \in \Sigma \,\}$, the trace monoid $M(D)$ presented by (Σ, D) is the free commutative monoid generated by Σ. Thus, $\mathrm{com}(E) = \bigcup_{w \in E} [w]_D$ is simply the linearization of the context-free trace language $\varphi_D(E)$. Hence, it follows from Theorem 4.6 that this language is accepted by a PD-CD-R(1)-system $\mathcal{M}'$. In fact, the system $\mathcal{M}'$ can effectively be constructed from the pushdown automaton B, and therewith from the given PD-CD-R(1)-system $\mathcal{M}$. This yields the following effective closure property.

Corollary 5.3. *The language classes* $\mathcal{L}(\mathsf{OC\text{-}CD\text{-}R(1)})$ *and* $\mathcal{L}(\mathsf{PD\text{-}CD\text{-}R(1)})$ *are effectively closed under the operation of taking the commutative closure.*

Also it is easily seen that these language classes are closed under *disjoint shuffle*, that is, if $L_1 \subseteq \Sigma^*$ and $L_2 \subseteq \Gamma^*$ are languages in $\mathcal{L}(\mathsf{OC\text{-}CD\text{-}R(1)})$ or $\mathcal{L}(\mathsf{PD\text{-}CD\text{-}R(1)})$, where $\Sigma \cap \Gamma = \emptyset$, then the shuffle of L_1 and L_2 is also in this language class. On the other hand, the following questions remain currently open.

(1) Are the language classes $\mathcal{L}(\mathsf{OC\text{-}CD\text{-}R(1)})$ and $\mathcal{L}(\mathsf{PD\text{-}CD\text{-}R(1)})$ closed under concatenation? It may be possible to carry the construction for stl-det-local-CD-R(1)-systems from [22] over to PD-CD-R(1)-systems, but currently it is not clear how to accommodate the pushdown operations in this construction.
(2) Are the language classes $\mathcal{L}(\mathsf{OC\text{-}CD\text{-}R(1)})$ and $\mathcal{L}(\mathsf{PD\text{-}CD\text{-}R(1)})$ closed under Kleene-star and Kleene-plus?
(3) Are they closed under reversal, ε-free morphisms, or inverse morphisms?

Finally we take a short look at decision problems for OC-CD-R(1)- and PD-CD-R(1)-systems. The membership problem for a language accepted by such a CD-system is obviously decidable nondeterministically in linear space and quadratic time. Further, from Theorem 3.8 and Corollary 3.9 it follows immediately that the emptiness problem and the finiteness problem are decidable for these classes of CD-systems. On the other hand, from the corresponding results for stl-det-local-CD-R(1)-systems in [22] it follows that the regularity problem, the inclusion problem, and the equivalence problem are all undecidable in general for OC-CD-R(1)- and PD-CD-R(1)-systems.

6. Concluding remarks

We have presented a class of automata (or rather, systems of automata) that accept all linearizations of context-free trace languages. In fact, we have even obtained a characterization of the linearizations of one-counter and context-free trace languages in terms of our model. In comparison to Zielonka's asynchronous automata for recognizable trace languages, our model has certainly the disadvantage that it just accepts words, that is, linearizations of traces, and not traces themselves. Thus, it is not a *concurrent model*. However, we see from the proof of Theorem 4.6 that the PD-CD-R(1)-system $\mathcal{M}$ that is constructed from a given context-free grammar G and a dependency relation D on Σ performs essentially the same computations for all linearizations of a given trace $[w]_D$ in the trace language defined by $L(G)$.

Further, the PD-CD-R(1)-system $\mathcal{M}$ is obtained in polynomial time from G and D (provided G is already in Greibach normal form). This contrasts with the situation for Zielonka's asynchronous automata, where the best currently known algorithm yields an asynchronous automaton that is polynomial in the size of the given word automaton and exponential in the number of concurrent processes [10]. Finally, the constructions in the proofs of Theorems 4.6 and 4.11 show that the

PD-CD-R(1)-systems yield an effective calculus for the class of linearizations of context-free trace languages: From PD-CD-R(1)-systems $\mathcal{M}_1$ and $\mathcal{M}_2$ for the linearizations of two context-free trace languages S_1 and S_2 over $M(D)$, we can effectively construct PD-CD-R(1)-systems for the linearizations of the context-free trace languages $S_1 \cup S_2$, $S_1 \cdot S_2$, and S_1^*.

References

[1] I. Aalbersberg and G. Rozenberg, Theory of traces. *Theoret. Comput. Sci.* **60** (1988) 1–82.

[2] J. Autebert, J. Berstel and L. Boasson, Context-free languages and pushdown automata, in *Handbook of Formal Languages, Word, Language, Grammar*, edited by G. Rozenberg and A. Salomaa. **1** (1997) 111–174.

[3] J. Berstel, *Transductions and Context-Free Languages*, *Leitfäden der angewandten Mathematik und Mechanik* **38** (1979).

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

[5] H. Bordihn, M. Holzer and M. Kutrib, Input reversals and iterated pushdown automata: a new characterization of Khabbaz geometric hierarchy of languages, in *Proc. of DLT 2004,* edited by C.S. Calude, E. Calude and M.J. Dinneen. *Lect. Notes Comput. Sci.* **3340** (2004) 102–113.

[6] P. Cartier and D. Foata, Problèmes Combinatoires de Commutation et Réarrangements. *Lect. Notes Comput. Sci.* **85** (1969).

[7] E. Csuhaj-Varjú, J. Dassow, J. Kelemen and G. Păun, *Grammar Systems. A Grammatical Approach to Distribution and Cooperation*, edited by Gordon and Breach. London (1994).

[8] E. Csuhaj-Varjú, C. Martín-Vide and V. Mitrana, Multiset automata, in *Multiset Processing*, edited by C.S. Calude, G. Păun, G. Rozenberg and A. Salomaa. *Lect. Notes Comput. Sci.* **2235** (2001) 69–83.

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

[10] B. Genest, H. Gimbert, A. Muscholl and I. Walukiewicz, Optimal Zielonka-type construction of deterministic asynchronous automata, in *Proc. of ICALP 2010, Part. II*, edited by S. Abramsky, C. Gavoille, C. Kircher, F. Meyer auf der Heide and P.G. Spirakis. *Lect. Notes Comput. Sci.* **6199** (2010) 52–63.

[11] M. Harrison, *Introduction to Formal Language Theory*. Addison-Wesley, Reading, MA (1978).

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

[13] P. Jančar, F. Mráz, M. Plátek and J. Vogel, Restarting automata, in *Proc. of FCT 1995*, edited by H. Reichel. *Lect. Notes Comput. Sci.* **965** (1995) 283–292.

[14] P. Jančar, F. Moller and Z. Sawa, Simulation problems for one-counter machines, in *Proc. of SOFSEM'99*, edited by J. Pavelka, G. Tel and M. Bartošek. *Lect. Notes Comput. Sci.* **1725** (1999) 404–413.

[15] M. Kudlek, P. Totzke and G. Zetzsche, Multiset pushdown automata. *Fund. Inform.* **93** (2009) 221–233.

[16] M. Kutrib, H. Messerschmidt and F. Otto, On stateless two-pushdown automata and restarting automata, in *Proc. of AFL 2008,* edited by E. Csuhaj-Varjú and Z. Ésik. Hungarian Academy of Sciences (2008) 257–268.

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

[18] A. Mazurkiewicz, *Concurrent program schemes and their interpretations*, *DAIMI Rep. PB.* Aarhus University, Aarhus **78** (1977).

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

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

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

[22] B. Nagy and F. Otto, *On CD-systems of stateless deterministic R-automata with window size one.* Kasseler Informatikschriften, Fachbereich Elektrotechnik/Informatik, Universität Kassel (2010).
`https://kobra.bibliothek.uni-kassel.de/handle/urn:nbn:de:hebis:34-2010042732682`

[23] B. Nagy and F. Otto, An automata-theoretical characterization of context-free trace languages, in *Proc. of SOFSEM 2011: Theory and Practice of Computer Science*, edited by I. Černá, T. Gyimóthy, J. Hromkovič, K. Jefferey, R. Královič, M. Vukolič and S. Wolf. *Lect. Notes Comput. Sci.* **6543** (2011) 406–417.

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

[25] F. Otto, Restarting automata, in *Recent Advances in Formal Languages and Applications*, *Studies in Computational Intelligence*, edited by Z. Ésik, C. Martin-Vide and V. Mitrana. **25** (2006) 269–303.

[26] W. Zielonka, Note on finite asynchronous automata. *RAIRO–Inf. Theor. Appl.* **21** (1987) 99–135.

Communicated by S. Crespi-Reghizzi.
Received March 3, 2011. Accepted August 8, 2011.