CD-Systems of Stateless Deterministic R(1)-Automata Governed by an External Pushdown Store*

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

[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

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

1 Introduction

Cooperating distributed systems (CD-systems) of restarting automata have been defined in [11], and in [12] various types of deterministic CD-systems of restarting automata have been studied. As expected CD-systems are much more expressive than their component automata themselves. For example, already the marked copy language $L_{\text{copy}} = \{ wcw \mid w \in \{a, b\}^* \}$ is accepted by a CD-system consisting of only two deterministic R-automata, although this language is not even growing context-sensitive, that is, it is not even accepted by any deterministic RRWW-automaton (see, e.g., [16]). On the other hand, stateless restarting automata, that is, restarting automata with only a single state, have been introduced and studied in [9, 10]. 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 [13] we introduced CD-systems of stateless deterministic restarting automata that have a read/write window of size 1 only. The restarting automata

* 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.

of this type have a severely restricted expressive power. However, by combining several such automata into a CD-system we obtain a device that is suprizingly expressive. In fact, in mode $= 1$ these systems accept a class of semi-linear languages that properly contains all rational trace languages. In fact, we even obtained a characterization of the rational trace languages in terms of a particular type of these CD-systems. Further, the class of languages that are accepted by mode $= 1$ computations of these CD-systems is closed under union, product, Kleene star, commutative closure, and disjoint shuffle, but it is not closed under intersection with regular languages, complementation, or ε-free morphisms. In addition, for these CD-systems the emptiness problem and the finiteness problem are easily solvable, while the regularity problem, the inclusion problem, and the equivalence problem are undecidable in general.

Here we extend these CD-systems by an external pushdown store that is used to determine the successor of the current automaton. When the active automaton performs a delete operation, then one of its successor automata is chosen based on the symbol deleted and on the topmost symbol on this pushdown store. In addition, after the successor has been chosen the pushdown content is modified by either erasing the topmost symbol, or by replacing it by a symbol or a word of length 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 $\mathsf{R}(1)$-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 [13] can be interpreted as *finite-state acceptors with translucent letters* (see [15]). 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.

This paper is structured as follows. In Section 2 we restate in short the definition of the CD-systems of stateless deterministic $\mathsf{R}(1)$-automata and their main properties from [13]. Then, in Section 3, we define the CD-systems of stateless deterministic $\mathsf{R}(1)$-automata that are governed by an external pushdown store (the so-called $\mathsf{PD\text{-}CD\text{-}R}(1)$-systems). We also consider the special case of these CD-systems when the pushdown is a counter (the so-called $\mathsf{OC\text{-}CD\text{-}R}(1)$-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 $\mathcal{L}_{=1}(\mathsf{stl\text{-}det\text{-}local\text{-}CD\text{-}R}(1))$ of languages that are accepted by CD-systems of stateless deterministic $\mathsf{R}(1)$-automata. In Section 4 we then study one-counter and context-free trace languages. We will see that our $\mathsf{PD\text{-}CD\text{-}R}(1)$-systems accept a proper superclass of the context-free trace languages, and the $\mathsf{OC\text{-}CD\text{-}R}(1)$-systems accept a proper superclass of the one-counter trace languages. However, we also provide characterizations of these classes of trace languages in terms of our CD-systems.

The paper closes with Section 5, which contains some preliminary closure and non-closure results and several open problems.

2 Stateless Deterministic R(1)-Automata

Stateless types of restarting automata were introduced in [9]. 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 u 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 head scans 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:

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

Thus, Σ_1 is the set of letters that M just moves across, Σ_2 is the set of letters that M deletes, Σ_3 is the set of letters which cause M to accept, and Σ_4 is the set of letters on which M will get stuck. It has been shown in [13] that the language $L(M)$ can be characterized as

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

Cooperating distributed systems of restarting automata were introduced and studied in [11]. Here we only consider *cooperating distributed systems of stateless deterministic* R(1)-*automata* (or stl-det-local-CD-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 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 $i \in I$. Actually, in [13] 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.

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 machine M_{i_l} accepts. Should at some stage the chosen machine M_{i_l} be unable to execute a cycle or to accept, then the computation fails. By $L_{=1}(\mathcal{M})$ we denote the language that the 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-R(1)) we denote the class of languages that are accepted by mode = 1 computations of stl-det-local-CD-R(1)-systems.

Example 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:

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

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 $L_{abc} := \{ w \in \{a, b, c\}^* \mid |w|_a = |w|_b = |w|_c \geq 1 \}$.

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 $1 \leq i \leq n$. In [13] the following results were established.

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

(b) $\mathcal{L}_{=1}(\mathsf{stl\text{-}det\text{-}local\text{-}CD\text{-}R(1)})$ *only contains languages that are semi-linear, that is, it only contains languages with semi-linear Parikh image.*

(c) $\mathcal{L}_{=1}(\mathsf{stl\text{-}det\text{-}local\text{-}CD\text{-}R(1)})$ *properly contains the class of all rational trace languages.*

As the deterministic linear language $L = \{\, a^n b^n \mid n \geq 0 \,\}$ does not contain a regular sublanguage that is letter-equivalent to the language itself, we see from (a) that this language is not accepted by any stl-det-local-CD-R(1)-system working in mode $= 1$. Together with Example 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. Here DLIN denotes the class of *deterministic linear languages*, which is the class of languages that are accepted by deterministic one-turn pushdown automata, LIN is the class of *linear languages*, DOCL and OCL denote the classes of *deterministic one-counter languages* and *one-counter languages* (see below), and DCFL and CFL denote the classes of *deterministic context-free languages* and *context-free languages.*

For technical reasons the following normal form was introduced in [13] for stl-det-local-CD-R(1)-systems, and it was shown 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 $L_{=1}(\mathcal{M}') = L_{=1}(\mathcal{M})$.

Definition 1. *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* $\Sigma_1^{(i)}, \Sigma_2^{(i)}, \Sigma_3^{(i)}, \Sigma_4^{(i)}$ *is the partitioning of alphabet* Σ *for the automaton* M_i *as described above:*

$$(1.)\ |\Sigma_2^{(i)}| \leq 1,\ (2.)\ \delta_i(\text{¢}) = \mathsf{MVR} \textit{ and } \Sigma_3^{(i)} = \emptyset,\ (3.)\ \Sigma_2^{(i)} = \emptyset \textit{ iff } \delta_i(\$) = \mathsf{Accept}.$$

In [14] closure properties and algorithmic properties are presented for these CD-systems.

3 CD-Systems with an External Pushdown Store

A *pushdown CD-system of stateless deterministic* R(1)-*automata*, PD-CD-R(1)-system for short, consists of a CD-system of stateless deterministic R(1)-automata and an external pushdown store. Formally, it 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 $(i, ¢w\$, \alpha)$, where $i \in I$ is the index of the active component automaton M_i, the word $¢w\$$ ($w \in \Sigma^*$) is a restarting configuration of M_i, and the word $\alpha \in \bot \cdot \Gamma^*$ is the current content of the pushdown store with the first symbol of α at the bottom and the last symbol of α at the top. For $w \in \Sigma^*$, an *initial configuration* of $\mathcal{M}$ on input w has the form $(i_0, ¢w\$, \bot)$ for any $i_0 \in I_0$, and an *accepting configuration* has the form $(i, \mathsf{Accept}, \bot)$.

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 $i \in I$, $w \in \Sigma^*$, $\alpha \in \bot \cdot \Gamma^*$, $A \in \Gamma$, and for each $i \in I$, $\Sigma_1^{(i)}$, $\Sigma_2^{(i)}$, and $\Sigma_3^{(i)}$ are the subsets of Σ that correspond to the automaton M_i (see Section 2):

(1) $(i, ¢w\$, \alpha A) \Rightarrow_{\mathcal{M}} (j, ¢w'\$, \alpha\eta)$ if $\exists u \in {\Sigma_1^{(i)}}^*, a \in \Sigma_2^{(i)}, v \in \Sigma^*$ 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 $\exists u \in {\Sigma_1^{(i)}}^*, a \in \Sigma_2^{(i)}, v \in \Sigma^*$ 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 $\exists u \in {\Sigma_1^{(i)}}^*, a \in \Sigma_3^{(i)}, v \in \Sigma^*$ such that $w = uav,$ or $w \in {\Sigma_1^{(i)}}^*$ and $\delta_i(\$) = \mathsf{Accept}$.

By $\Rightarrow_{\mathcal{M}}^*$ we denote the *computation relation* of $\mathcal{M}$, which is simply 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,

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

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 $\mathcal{L}(\mathsf{PD\text{-}CD\text{-}R(1)})$ we denote the class of languages that are accepted by PD-CD-R(1)-systems, and $\mathcal{L}(\mathsf{OC\text{-}CD\text{-}R(1)})$ denotes the class of languages that are accepted by OC-CD-R(1)-systems.

Example 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 1 (a), L is not accepted by any stl-det-local-CR-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:

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

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

- $\sigma_a = \{a, b\}$, $\sigma_b = \{c\}$, $\sigma_c = \{b, +\}$, and $\sigma_+ = \{+\}$,
- $\Gamma = \{C\}$,
- $I_0 = \{a, +\}$, 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 $(+, ¢ \cdot \$, \bot)$. Hence, we see that $L(\mathcal{M}) = L$ holds.

Thus, already the language class $\mathcal{L}(\mathsf{OC\text{-}CD\text{-}R(1)})$ contains a language that is neither context-free nor accepted by any stl-det-local-CD-R(1)-system. Next we will show that the class of languages that are accepted by the latter type of CD-systems is contained in $\mathcal{L}(\mathsf{OC\text{-}CD\text{-}R(1)})$.

Proposition 2. $\mathcal{L}_{=1}(\mathsf{stl\text{-}det\text{-}local\text{-}CD\text{-}R(1)}) \subsetneq \mathcal{L}(\mathsf{OC\text{-}CD\text{-}R(1)})$.

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 $i \in I$:

$$\begin{array}{ll} \delta(i, a, \bot) = \{\, (j, \bot) \mid j \in \sigma_i \,\} & \text{for all } a \in \Sigma_2^{(i)}, \\ \delta(i, a, \bot) = \emptyset & \text{for all } a \in \Sigma \smallsetminus \Sigma_2^{(i)}. \end{array}$$

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. □

On the other hand, PD-CD-R(1)-systems accept all context-free languages.

Proposition 3. $\mathsf{CFL} \subsetneq \mathcal{L}(\mathsf{PD\text{-}CD\text{-}R(1)})$.

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., [7]). 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:

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

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 $a \in \Sigma$:

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

and δ yields the empty set for all other values. Then, for all $w \in \Sigma^*$ and all $a \in \Sigma$,

$$\begin{array}{lll} wa \in L \text{ iff } S \Rightarrow_G^+ wA \Rightarrow_G wa & & \\ \quad \text{iff } (S, ¢ \cdot wa \cdot \$, \bot) & \Rightarrow_{\mathcal{M}}^* & (A, ¢ \cdot a \cdot \$, \bot A) \Rightarrow_{\mathcal{M}} (+, ¢ \cdot \$, \bot) \\ & \Rightarrow_{\mathcal{M}} & (+, \mathsf{Accept}, \bot). \end{array}$$

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

If the given context-free language includes the empty word, we can apply the above construction to the language $L \smallsetminus \{\varepsilon\}$. Then the resulting $\mathsf{PD\text{-}CD\text{-}R(1)}$-system will accept this language. By adding the component $+$ to the set of initial components, we obtain a $\mathsf{PD\text{-}CD\text{-}R(1)}$-system for the language L. This yields the intended inclusion, which is proper by Example 2. □

Next we consider the so-called *one-counter automata* and the class of languages accepted by them. However, 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 [8] (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 is 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 that are accepted by one-counter automata. It is well-known that $\mathsf{REG} \subsetneq \mathsf{OCL} \subsetneq \mathsf{CFL}$ holds.

Proposition 4. $\mathsf{OCL} \subsetneq \mathcal{L}(\mathsf{OC\text{-}CD\text{-}R(1)})$.

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

1. $I = (Q \times \{=, >\}) \cup \{+\}$,
2. $I_0 = \{(q_0, =), +\}$,
3. $\sigma_{(q,>)} = \sigma_{(q,=)} = \sigma_+ = I$ for all $q \in Q$,
4. the stateless deterministic $\mathsf{R(1)}$-automata $M_{(q,>)}$, $M_{(q,=)}$ $(q \in Q)$, and M_+ are defined as follows:

$$\begin{array}{lll}
(1)\ \delta_{(q,=)}(¢) = \mathsf{MVR}, & \\
(2)\ \delta_{(q,=)}(a) = \varepsilon & \text{if } \delta_{\mathcal{A}}(q, a, \bot) \text{ is defined,} \\
(3)\ \delta_{(q,>)}(¢) = \mathsf{MVR}, & \\
(4)\ \delta_{(q,>)}(a) = \varepsilon & \text{if } \delta_{\mathcal{A}}(q, a, C) \text{ is defined,} \\
(5)\ \delta_{+}(¢) \quad = \mathsf{MVR}, & \\
(6)\ \delta_{+}(\$) \quad = \mathsf{Accept}, &
\end{array}$$

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

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

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 2. □

Definition 2. *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* $i \in I$, $\Sigma_1^{(i)}, \Sigma_2^{(i)}, \Sigma_3^{(i)}, \Sigma_4^{(i)}$ *is the partitioning of alphabet* Σ *for the automaton* M_i *as described in Section 2:*

$$\begin{aligned}
&(1)\ \exists\, i_+ \in I : \delta_{i_+}(¢) = \mathsf{MVR}, \delta_{i_+}(\$) = \mathsf{Accept},\ \textit{and}\ \Sigma_4^{(i_+)} = \Sigma; \\
&(2)\ \forall i \in I \smallsetminus \{i_+\} : \delta_i(¢) = \mathsf{MVR}, |\Sigma_2^{(i)}| = 1, \Sigma_3^{(i)} = \emptyset,\ \textit{and}\ \delta_i(\$) = \emptyset.
\end{aligned}$$

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 1. *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 a* OC-CD-R(1)*-system, then* $\mathcal{M}'$ *can be constructed to be a* 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. First we split every component automaton M_i into $|\Sigma_2^{(i)}| + 1$ many parts, $M_i^{(a)}$ for $a \in \Sigma_2^{(i)}$, and $M_i^{(+)}$, where the former is responsible for executing the cycles of M_i in which an occurrence of the letter a is deleted, while the latter takes care of the accepting tail computations of M_i. In detail, for each $a \in \Sigma_2^{(i)}$, and all $b, c \in \Sigma$,

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

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

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

Observe, however, that the successor relations $\sigma_i^{(+)}$ are never used in any computation. We take

$$\hat{\mathcal{M}} = (\hat{I}, \Sigma, (M_i^{(a)}, \sigma_i^{(a)})_{i \in I, a \in \Sigma_2^{(i)}} \cup (M_i^{(+)}, \sigma_i^{(+)})_{i \in I}, \Gamma, \perp, \hat{I}_0, \hat{\delta}),$$

where $\hat{I} = \{\, i^{(a)}, i^{(+)} \mid i \in I,\, a \in \Sigma_2^{(i)} \,\}$, and $\hat{I}_0 = \{\, i^{(a)}, i^{(+)} \mid i \in I_0,\, a \in \Sigma_2^{(i)} \,\}$. Finally, the successor relation $\hat{\delta} : (\hat{I} \times \Sigma \times (\Gamma \cup \{\perp\}) \to 2^{\hat{I} \times (\Gamma \cup \{\perp\})^*}$ is defined as follows, where $i \in I$, $a, b \in \Sigma$, and $A \in \Gamma$:

$$\begin{aligned}
(1)\ \hat{\delta}(i^{(a)}, a, A) &= \{\, (j^{(c)}, \alpha) \mid (j, \alpha) \in \delta(i, a, A), c \in \Sigma_2^{(j)} \,\} \\
&\quad \cup \{\, (j^{(+)}, \varepsilon) \mid (j, \varepsilon) \in \delta(i, a, A) \,\}, \\
(2)\ \hat{\delta}(i^{(a)}, a, \perp) &= \{\, (j^{(c)}, \alpha) \mid (j, \alpha) \in \delta(i, a, \perp), c \in \Sigma_2^{(j)} \,\} \\
&\quad \cup \{\, (j^{(+)}, \perp) \mid (j, \perp) \in \delta(j, a, \perp) \,\},
\end{aligned}$$

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 $\perp$ 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 $\perp$ (see (2)) or when it could just now have been reduced to $\perp$ (see (1)). It follows easily that $L(\hat{\mathcal{M}}) = L(\mathcal{M})$.

In order to obtain the intended system in normal form, we need 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, $\delta'_+(¢) = \mathsf{MVR}$ and $\delta'_+(\$) = \mathsf{Accept}$. Now we need to distinguish three cases.

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

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

Then in combination with M'_+, $M_i'^{(+)}$ accepts all words from Σ^*. We adjust the successor relation by defining

$$\delta'(i'^{(+)}, a, \bot) = \{(i'^{(+)}, \bot), (+, \bot)\}.$$

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

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

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

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

where $M_i''^{(+)}$ is the only successor of $M_i'^{(+)}$. Then together with M'_+ they accept the same words as $M_i^{(+)}$, but an accept instruction is only executed on the \$-symbol. In this case we must modify the successor relation $\hat{\delta}$ as follows:

$$\begin{aligned} \delta'(i'^{(+)}, a, \bot) &= \{(i''^{(+)}, \bot), (+, \bot)\} && \text{for all } a \in \Sigma_3^{(i)}, \\ \delta'(i''^{(+)}, a, \bot) &= \{(i''^{(+)}, \bot), (+, \bot)\} && \text{for all } a \in \Sigma. \end{aligned}$$

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

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

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

$$\begin{aligned} \delta'(i'^{(+)}, a, \bot) &= \{(i''^{(+)}, \bot), (+, \bot)\} && \text{for all } a \in \Sigma_3^{(i)}, \\ \delta'(i''^{(+)}, a, \bot) &= \{(i''^{(+)}, \bot), (+, \bot)\} && \text{for all } a \in \Sigma, \\ \delta'(\hat{i}^{(+)}, a, \bot) &= \{(\hat{i}^{(+)}, \bot), (+, \bot)\} && \text{for all } a \in \Sigma_1^{(i)}. \end{aligned}$$

Finally we again split each component automaton that contains more than one rewrite instruction into several automata, one for each letter that is deleted by a rewrite instruction. 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 $\mathcal{M}$.

We see from the description above 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 1. □

We have seen that the language class $\mathcal{L}(\mathsf{PD\text{-}CD\text{-}R(1)})$ 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 1. *Each language $L \in \mathcal{L}(\mathsf{PD\text{-}CD\text{-}R(1)})$ contains a context-free sublanguage 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 1 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 $i \in I_r := I \smallsetminus \{+\}$, M_i does not execute any accept instructions and $|\Sigma_2^{(i)}| = 1$. To simplify the notation in the following we denote the letter $a \in \Sigma_2^{(i)}$ simply by $a_{(i)}$.

From $\mathcal{M}$ we construct a pushdown automaton $P = (Q, \Sigma, \Gamma, q_0, \bot, \delta_P, F)$ as follows:

- $Q = I \cup \{q_0\}$, where q_0 is a new state,
- $F = \{+\}$, and
- the transition relation δ_P is defined as follows for all $i \in I_r$, $a \in \Sigma$, and $A \in \Gamma$:

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

Then $E = L(P)$ is a context-free language. It remains to show that it is a sublanguage of L that is letter-equivalent to L.

Claim 1. For all $i_0 \in I$, all $w \in \Sigma^*$, and all $\alpha \in \Gamma^*$, if $(i_0, ¢\cdot w\cdot \$, \bot\alpha) \Rightarrow_{\mathcal{M}} (i_1, ¢\cdot w_1 \cdot \$, \bot\alpha_1) \Rightarrow_{\mathcal{M}} \cdots \Rightarrow_{\mathcal{M}} (i_s, ¢\cdot w_s \cdot \$, \bot\alpha_s) \Rightarrow_{\mathcal{M}} (+, ¢\cdot \$, \bot) \Rightarrow_{\mathcal{M}} (+, \mathsf{Accept}, \bot)$ 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 there are two cases. Either $i_0 = +$, and then $w = \varepsilon$ and $\alpha = \varepsilon$, too, or $i_0 \in I_r$, and then $w = a_{(i_0)}$, $|\alpha| \leq 1$, and $(+, \bot) \in \delta(i_0, a_{(i_0)}, \bot)$, if $\alpha = \varepsilon$, or $(+, \varepsilon) \in \delta(i_0, a_{(i_0)}, A)$, if $\alpha = A \in \Gamma$. In the first case we take $z = \varepsilon$, which implies that $(i_0, z, \bot\alpha) = (+, \varepsilon, \bot)$, and in the latter case 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 $(i_0, ¢ \cdot w \cdot \$, \bot\alpha) \Rightarrow_{\mathcal{M}} (i_1, ¢ \cdot w_1 \cdot \$, \bot\alpha_1)$, w has a factorization $w = uav$ such that $u \in {\Sigma_1^{(i_0)}}^*$, $a = a_{(i_0)}$, and $w_1 = uv$, $\alpha = \varepsilon$, and $\alpha_1 = \eta$ such that $(i_1, \eta) \in \delta(i_0, a_{(i_0)}, \bot)$, 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

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

holds. By Claim 1 it follows 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.

The proof of Theorem 1 is now completed by establishing the following claim.

Claim 2. $E \subseteq L$.

Proof. Let $z \in E$, that is, $(q_0, z, \bot) \vdash_P^* (+, \varepsilon, \bot)$. We proceed by induction on $n = |z|$. If $n = 0$, then $z = \varepsilon$, and hence, $(+, \bot) \in \delta_P(q_0, \varepsilon, \bot)$. From the definition of δ_P we conclude that $+ \in I_0$, which implies that $z = \varepsilon \in L$.

If $n = 1$, then $z = a \in \Sigma$. Hence,

$$(q_0, z, \bot) = (q_0, a, \bot) \vdash_P (i, a, \bot) \vdash_P (+, \varepsilon, \bot)$$

for some $i \in I_0$ such that $(+, \bot) \in \delta_P(i, a, \bot) = \delta(i, a, \bot)$. Hence, $a = a_{(i)}$, and

$$(i, ¢ \cdot z \cdot \$, \bot) = (i, ¢ \cdot a_{(i)} \cdot \$, \bot) \Rightarrow_{\mathcal{M}} (+, ¢ \cdot \$, \bot) \Rightarrow_{\mathcal{M}} (+, \mathsf{Accept}, \bot)$$

is an accepting computation of $\mathcal{M}$ on input z, that is, $z \in L$.

If $n > 1$, then $z = az'$ for some $a \in \Sigma$ and $z' \in \Sigma^+$, and the accepting computation of P on input z has the following form:

$$(q_0, z, \bot) = (q_0, az', \bot) \vdash_P (i_0, az', \bot) \vdash_P (i_1, z', \bot\alpha) \vdash_P^* (+, \varepsilon, \bot)$$

for some $i_0 \in I_0$ such that $a = a_{(i_0)}$ and $(i_1, \bot\alpha) \in \delta(i_0, a, \bot)$. Thus, $\mathcal{M}$ can perform the following computational step:

$$(i_0, ¢ \cdot z \cdot \$, \bot) = (i_0, ¢ \cdot az' \cdot \$, \bot) \Rightarrow_{\mathcal{M}} (i_1, ¢ \cdot z' \cdot \$, \bot\alpha).$$

From the definition of δ_P we can conclude by induction that there exists a computation of $\mathcal{M}$ of the form

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

which implies that $z \in L$ holds. Hence, we see that E is indeed a subset of L. □

Together Claims 1 and 2 prove Theorem 1. □

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

Corollary 1. *Each language $L \in \mathcal{L}(\mathsf{OC\text{-}CD\text{-}R(1)})$ 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 a* OC-CD-R(1)*-system for L.*

As each context-free language has a semi-linear Parikh image, Theorem 1 has the following consequence.

Corollary 2. *The language class $\mathcal{L}(\mathsf{PD\text{-}CD\text{-}R(1)})$ 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 1 yields the following negative result.

Proposition 5. *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, but based on Corollary 1 we can show that it is not accepted by any OC-CD-R(1)-system.

Proposition 6. *The language $L_{pal} = \{\, wcw^R \mid w \in \{a, b\}^* \,\}$ is not accepted by any* OC-CD-R(1)*-system.*

Proof. By Corollary 1 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:

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

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 $ucu^R \in E$ and $vcv^R \in E$ 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

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 $I_D = (\Sigma \times \Sigma) \smallsetminus D$ is called the corresponding *independence relation.* Obviously, the relation I_D is irreflexive and symmetric. The dependency relation D (or rather its associated independence relation I_D) induces a binary relation $\equiv_D$ on Σ^* that is defined as the smallest congruence relation containing the set of pairs $\{ (ab, ba) \mid (a, b) \in I_D \}$. For $w \in \Sigma^*$, the congruence class of w mod $\equiv_D$ is denoted by $[w]_D$, that is, $[w]_D = \{ z \in \Sigma^* \mid w \equiv_D z \}$. These equivalence classes are called *traces*, and the factor monoid $M(D) = \Sigma^*/\equiv_D$ is a *trace monoid*. In fact, $M(D)$ is the *free partially commutative monoid* presented by (Σ, D) (see, e.g., [6]). 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., [6] Claim A in the proof of Prop. 6.2.2) will be useful in what follows.

Proposition 7. *For all words $x, y, u \in \Sigma^*$ and all letters $a \in \Sigma$, if $xay \equiv_D au$ and $|x|_a = 0$, then $(a, x) \in I_D$, $xay \equiv_D axy$, and $xy \equiv_D u$.*

A subset S of a trace monoid $M(D)$ is called *recognizable* if there exist a finite monoid N, a morphism $\alpha : M(D) \to N$, and a subset P of N such that $S = \alpha^{-1}(P)$ [3]. Accordingly, this property can be characterized as follows (see [6] Prop. 6.1.10).

Proposition 8. *Let $M(D)$ be the trace monoid presented by (Σ, D), and let $\varphi_D : \Sigma^* \to M(D)$ be the corresponding morphism. Then a set $S \subseteq M(D)$ is recognizable if and only if the language $\varphi_D^{-1}(S)$ is a regular language over Σ.*

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

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

Proposition 9. *Let $M(D)$ be the trace monoid presented by (Σ, D), and let $\varphi_D : \Sigma^* \to M(D)$ be the corresponding morphism. Then a set $S \subseteq M(D)$ is rational if and only if there exists a regular language R over Σ such that $S = \varphi_D(R)$.*

By $\mathsf{RAT}(M(D))$ we denote the set of rational subsets of $M(D)$. Concerning the relationship between the recognizable subsets of $M(D)$ and the rational subsets of $M(D)$ the following results are known (see, e.g., [6]).

Proposition 10. *For each trace monoid $M(D)$, $\mathsf{REC}(M(D)) \subsetneq \mathsf{RAT}(M(D))$, and these two sets are equal if and only if $I_D = \emptyset$.*

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

We call a language $L \subseteq \Sigma^*$ a *rational trace language*, if there exists a dependency relation D on Σ such that $L = \varphi_D^{-1}(S)$ for a rational subset S of the trace monoid $M(D)$ presented by (Σ, D). From Proposition 9 it follows that L is a rational trace language if and only if there exist a trace monoid $M(D)$ and a regular language $R \subseteq \Sigma^*$ such that $L = \varphi_D^{-1}(\varphi_D(R)) = \bigcup_{w \in R}[w]_D$. By $\mathcal{LRAT}(D)$ we denote the set of rational trace languages $\varphi_D^{-1}(\mathsf{RAT}(M(D)))$, and $\mathcal{LRAT}$ is the class of all rational trace languages. In [13] the following result on rational trace languages was established.

Theorem 2. $\mathcal{LRAT} \subsetneq \mathcal{L}_{=1}(\mathsf{stl\text{-}det\text{-}local\text{-}CD\text{-}R(1)})$, *that is, if $M(D)$ be 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 of traces $S \subseteq M(D)$.*

Here we are interested in more general trace languages. A language $L \subseteq \Sigma^*$ is called a *one-counter trace language*, if there exist a dependency relation D on Σ and a one-counter language $R \subseteq \Sigma^*$ such that $L = \varphi_D^{-1}(\varphi_D(R)) = \bigcup_{w \in R}[w]_D$. Analogously, a language $L \subseteq \Sigma^*$ is called a *context-free trace language*, if there exist a dependency relation D on Σ and a context-free language $R \subseteq \Sigma^*$ such that $L = \varphi_D^{-1}(\varphi_D(R)) = \bigcup_{w \in R}[w]_D$ [1,4]. By $\mathcal{LOC}(D)$ we denote the set of one-counter trace languages obtained from (Σ, D), and $\mathcal{LOC}$ is the class of all one-counter trace languages. Further, by $\mathcal{LCF}(D)$ we denote the set of context-free trace languages obtained from (Σ, D), and $\mathcal{LCF}$ is the class of all context-free trace languages. The next theorem states that all context-free trace languages are accepted by $\mathsf{PD\text{-}CD\text{-}R(1)}$-systems.

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

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

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 $L \subseteq \Sigma^*$ be a context-free trace language, that is, there exists a context-free language R over Σ such that $L = \varphi_D^{-1}(\varphi_D(R))$. As R is context-free, there exists a grammar $G = (V, \Sigma, S, P)$ in quadratic Greibach normal form for $R' = R \smallsetminus \{\varepsilon\}$. From G we construct a $\mathsf{PD\text{-}CD\text{-}R(1)}$-system $\mathcal{M} = (I, \Sigma, (M_i, \sigma_i)_{i \in I}, V, \perp, I_0, \delta)$ as follows (cf. the proof of Proposition 3):

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

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

- the sets of successor indices 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 $a \in \Sigma$:

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

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 $\varepsilon \in L$. From the definition of $\mathcal{M}$ we see that in this case $+ \in I_0$, which implies that $w \in L(\mathcal{M})$.

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 $w_0 = u \in R$. Thus, w_0 is generated by the grammar G, and hence, it follows as in the proof of Proposition 3 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:

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

or

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

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:

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

or

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

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

$$\begin{array}{rl} ((S,a_n), ¢ \cdot w \cdot \$, \bot) \Rightarrow_{\mathcal{M}} & ((A_{n-1},a_{n-1}), ¢ \cdot w_{n-1} \cdot \$, \bot\alpha_{n-1}) \\ \Rightarrow_{\mathcal{M}} & ((A_{n-2},a_{n-2}), ¢ \cdot w_{n-2} \cdot \$, \bot\alpha_{n-2}) \\ \Rightarrow_{\mathcal{M}}^{*} & ((A_2,a_2), ¢ \cdot w_2 \cdot \$, \bot\alpha_2) \\ \Rightarrow_{\mathcal{M}} & ((A_1,a_1), ¢ \cdot w_1 \cdot \$, \bot\alpha_1) \\ \Rightarrow_{\mathcal{M}} & (+, ¢ \cdot \$, \bot) \\ \Rightarrow_{\mathcal{M}} & (+, \mathsf{Accept}, \bot) \end{array}$$

be an accepting computation of $\mathcal{M}$ on input $w = w_n$. 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, $w = w_1 = a_1 \in R$.

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

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

$$((A_1,a_1), ¢ \cdot w_1 \cdot \$, \bot\alpha_1) \Rightarrow_{\mathcal{M}} (+, ¢ \cdot \$, \bot),$$

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 $\alpha_1^R = A_1 \Rightarrow_G a_1 = u_1 = w_1$.

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

$$((A_{i+1},a_{i+1}), ¢ \cdot w_{i+1} \cdot \$, \bot\alpha_{i+1}) \Rightarrow_{\mathcal{M}} ((A_i,a_i), ¢ \cdot w_i \cdot \$, \bot\alpha_i).$$

Thus, $w_{i+1} = ua_{i+1}v$ and $w_i = uv$ for some $u \in \Sigma_1^{(A_{i+1},a_{i+1})^*}$ 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 $w_i = uv$, we see that $u_{i+1} = a_{i+1}u_i$ is letter-equivalent to $w_{i+1} = ua_{i+1}v$. Further, as $u \in \Sigma_1^{(A_{i+1},a_{i+1})^*}$, $(b, a_{i+1}) \in I_D$ for all letters occurring in u, which means that

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

This completes the inductive step.

Finally, from $((S, a_n), ¢ \cdot w \cdot \$, \bot) \Rightarrow_{\mathcal{M}} ((A_{n-1}, a_{n-1}), ¢ \cdot w_{n-1} \cdot \$, \bot\alpha_{n-1})$, we see that $w = u'a_nv'$ and $w_{n-1} = u'v'$ 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 see that

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

that is, $u_n \in R$. Further, $u_n = a_nu_{n-1} \equiv_D a_nw_{n-1} = a_nu'v' \equiv_D u'a_nv' = w$. 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 3. □

An analogous result also holds for one-counter languages.

Theorem 4. *Let $M(D)$ be the trace monoid presented by (Σ, D), where D is a dependency relation on the finite alphabet Σ. Then*

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

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

Proof. Let $L \subseteq \Sigma^*$ be a one-counter trace language, that is, there exists a one-counter language R over Σ such that $L = \varphi_D^{-1}(\varphi_D(R))$. As R is a one-counter language, 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 a OC-CD-R(1)-system $\mathcal{M} = (I, \Sigma, (M_i, \sigma_i)_{i \in I}, \{C\}, \bot, I_0, \delta)$ as follows (cf. the proof of Proposition 4):

1. $I = (Q \times \{=, >\} \times \Sigma) \cup \{+\}$,
2. $I_0 = \{\, (q_0, =, a) \mid a \in \Sigma \,\} \cup \{\, + \mid \varepsilon \in L \,\}$,
3. $\sigma_{(q,>,a)} = \sigma_{(q,=,a)} = \sigma_+ = I$ for all $q \in Q$ and all $a \in \Sigma$,

$$\begin{tikzcd}
 & & \mathcal{L}(\mathsf{PD\text{-}CD\text{-}R(1)}) \\
 & \mathcal{LCF} \arrow[ur] & \mathcal{L}(\mathsf{OC\text{-}CD\text{-}R(1)}) \arrow[u] \\
\mathsf{CFL} \arrow[ur] & \mathcal{LOC} \arrow[u] \arrow[ur] & \mathcal{L}_{=1}(\mathsf{stl\text{-}det\text{-}local\text{-}CD\text{-}R(1)}) \arrow[u] \\
\mathsf{OCL} \arrow[u] \arrow[ur] & \mathcal{LRAT} \arrow[u] \arrow[ur] & \\
\mathsf{REG} \arrow[u] \arrow[ur] & &
\end{tikzcd}$$

Fig. 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.

4. the stateless deterministic R(1)-automata $M_{(q,>,a)}$, $M_{(q,=,a)}$ $(q \in Q,\ a \in \Sigma)$, and M_+ are defined as follows:

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

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

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

while δ yields the empty set for all other values. As in the proof of Theorem 3 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 6 we see that the class $\mathcal{L}(\mathsf{OC\text{-}CD\text{-}R(1)})$ 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 [14] the language L' is accepted by a stl-det-local-CD-R(1)-system. However, L' is not a context-free trace language.

Proposition 11. *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 $c^n(ab)^n \cdot a^m \in L'$, 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, contradict the assumption $L' = \bigcup_{w\in R}[w]_D$, as $c^n a^{n+m} b^n \notin L'$. Thus, $(a,b),(b,a) \in D$ follows. □

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

It follows 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 $b^n a^n c^n \cdot a^m \in L'$, 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 11. □

Together with Proposition 6 this result we have the following consequences.

Corollary 3.
(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 context-free trace languages.

Definition 3. *Let $\mathcal{M} = (I, \Sigma, (M_i, \sigma_i)_{i\in I}, V, \bot, I_0, \delta)$ be a* PD-CD-R(1)*-system in strong normal form that satisfies the following condition:*

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

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

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

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

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

Theorem 5. *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 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 $\mathcal{M} = (I, \Sigma, (M_i, \sigma_i)_{i\in I}, V, \bot, I_0, \delta)$ be a PD-CD-R(1)-system in strong normal form on Σ. Assume that $\mathcal{M}$ satisfies condition $(*)$, and that the associated relation $I_{\mathcal{M}} = \bigcup_{i\in I}(\Sigma_1^{(i)} \times \Sigma_2^{(i)})$ is symmetric. Then the relation $D_{\mathcal{M}} = (\Sigma \times \Sigma) \smallsetminus I_{\mathcal{M}}$ 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 $w \in L_{=1}(\mathcal{M})$ 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 $\Sigma_2^{(i)} = \{a\}$.

2. If $(a,b) \in I_{\mathcal{M}}$, then $a \in \Sigma_1^{(i)}$ for all component automata M_i for which $\Sigma_2^{(i)} = \{b\}$ holds.
3. If $(a,b) \in I_{\mathcal{M}}$, then $(b,a) \in I_{\mathcal{M}}$, too, and hence, $b \in \Sigma_1^{(j)}$ for all component automata M_j for which $\Sigma_2^{(j)} = \{a\}$ holds.

Let $L = L(\mathcal{M})$. We claim that L is 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 $i \in I_r := I \smallsetminus \{+\}$, $a \in \Sigma$, and $A \in \Gamma$:

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

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 1. Thus, it follows that $R = L(B)$ is a sublanguage of $L(\mathcal{M})$ that is letter-equivalent to $L(\mathcal{M})$.

Let $w \equiv_{D_{\mathcal{M}}} u \in R$, and let $u = w_0, w_1, \ldots, w_n = w$ be a sequence of words such that, for each $i = 1, \ldots, n$, w_i is obtained from w_{i-1} by replacing a factor ab by ba for some pair of letters $(a,b) \in I_{\mathcal{M}}$. We now prove that $w_i \in L$ for all i by induction on i.

For $i = 0$ we have $w_0 = u \in R$, and so $w_0 \in L(\mathcal{M})$ by the considerations in the previous paragraph. 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 $(a,b) \in I_{\mathcal{M}}$. By our hypothesis $\mathcal{M}$ has an accepting computation for $w_i = xaby$, which is of one of the following two forms:

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

or

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

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, $\Sigma_2^{(i_1)} = \{a\} = \Sigma_2^{(j_3)}$ and $\Sigma_2^{(i_3)} = \{b\} = \Sigma_2^{(j_1)}$, 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 $b \in \Sigma_1^{(i_1)}$. Hence, in the former case we obtain the computation

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

while in the latter case we have the computation

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

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

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

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

$$\begin{array}{rlll}(i_n, ¢ \cdot w_n \cdot \$, \bot) & \Rightarrow_{\mathcal{M}} (i_{n-1}, ¢ \cdot w_{n-1} \cdot \$, \bot\alpha_{n-1}) & \Rightarrow_{\mathcal{M}} & (i_{n-2}, ¢ \cdot w_{n-2} \cdot \$, \bot\alpha_{n-2}) \\ & \Rightarrow_{\mathcal{M}} \quad \dots & \Rightarrow_{\mathcal{M}} & (i_1, ¢ \cdot w_1 \cdot \$, \bot\alpha_1) \\ & \Rightarrow_{\mathcal{M}} \quad (+, ¢ \cdot \$, \bot) & \Rightarrow_{\mathcal{M}} & (+, ¢ \cdot \$, \mathsf{Accept})\end{array}$$

be an accepting computation of $\mathcal{M}$ on input $w = w_n$. We claim that, for each $j = 1, \dots, n$, 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 $\Sigma_2^{(i_1)} = \{a_1\}$, and either $\alpha_1 = \bot$ and $(+, \bot) \in \delta(i_1, a_1, \bot)$, or $\alpha_1 = \bot A$ 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 $(i_1, w_1, \alpha_1) = (i_1, a_1, \alpha_1) \vdash_B (+, \varepsilon, \bot)$ 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, α_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

$$(i_{j+1}, ¢ \cdot w_{j+1} \cdot \$, \bot\alpha_{j+1}) \Rightarrow_{\mathcal{M}} (i_j, ¢ \cdot w_j \cdot \$, \bot\alpha_j).$$

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 $\Sigma_2^{(i_{j+1})} = \{a_{j+1}\}$, $\bot\alpha_{j+1} = \gamma A$ and $\bot\alpha_j = \gamma\eta$ for some $\gamma \in \Gamma^*$ and $A \in \Gamma \cup \{\bot\}$, and $(i_j, \eta) \in \delta(i_{j+1}, a_{j+1}, A)$. Also we see that $(x, a_{j+1}) \in I_{\mathcal{M}}$.

Again from the definition of B it follows that $(i_j, \eta) \in \delta_B(i_{j+1}, a_{j+1}, A)$, which implies that B can perform the computational step

$$(i_{j+1}, a_{j+1}u_j, \bot\alpha_{j+1}) = (i_{j+1}, a_{j+1}u_j, \gamma A) \vdash_B (i_j, u_j, \gamma\eta) = (i_j, u_j, \bot\alpha_j).$$

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 5. □

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 $\delta_B(q_0, a, \bot) = \{\, (j, \eta) \mid \exists i \in I_0 : (j, \eta) \in \delta(i, a, \bot) \,\}$ for all $a \in \Sigma$. Thus, we also have the following result.

Corollary 4. *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 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 3 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 3 and 5 together yield the following characterization.

Corollary 5. (a) *A language $L \subseteq \Sigma^*$ is 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 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 $L = L(\mathcal{M})$.*

5 Closure and Non-Closure Properties

As seen in Example 2 the language

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

is accepted by a 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. 5). This gives the following non-closure result.

Corollary 6. *The language classes* $\mathcal{L}(\mathsf{OC\text{-}CD\text{-}R(1)})$ *and* $\mathcal{L}(\mathsf{PD\text{-}CD\text{-}R(1)})$ *are not closed under intersection with regular languages.*

Next we consider the closure under Boolean operations.

Proposition 12.

(a) *The language classes* $\mathcal{L}(\mathsf{OC\text{-}CD\text{-}R(1)})$ *and* $\mathcal{L}(\mathsf{PD\text{-}CD\text{-}R(1)})$ *are closed under union.*
(b) *The language classes* $\mathcal{L}(\mathsf{OC\text{-}CD\text{-}R(1)})$ *and* $\mathcal{L}(\mathsf{PD\text{-}CD\text{-}R(1)})$ *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 a OC-CD-R(1)-system. Hence, Corollary 6 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* 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 (Theorem 1). Obviously, the commutative closure com(L) of L coincides with the commutative closure 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 context-free trace language $\varphi_D^{-1}(\varphi_D(E))$. Hence, it follows from Theorem 2 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 7. *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 product?

   Remark. It may be possible to carry the construction for stl-det-local-CD-R(1)-systems from [14] over to PD-CD-R(1)-systems. However, there is a problem with the pushdown store: whenever a system $\mathcal{M}_1$ deletes the last occurrence of a letter a from its tape, then the simulating automaton $\mathcal{M}'_1$ just remembers this by changing the corresponding index from 1 to d. However, this means that not only the erasing of the letter a is not performed, but also the corresponding operation on the pushdown store is not executed. Hence, from this point on the pushdown stores of $\mathcal{M}_1$ and $\mathcal{M}'_1$ differ. Is there a way to overcome this problem?
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?
4. Are they closed under ε-free morphisms?
5. Are they closed under 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 1 and Corollary 1 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 [14] 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.

References

1. I. Aalbersberg and G. Rozenberg. Theory of traces. *Theoretical Computer Science* 60 (1988) 1–82.
2. J. Autebert, J. Berstel, and L. Boasson. Context-free languages and pushdown automata. In: G. Rozenberg and A. Salomaa(eds.), *Handbook of Formal Languages, Vol. 1: Word, Language, Grammar*, Springer, Berlin, 1997, 111–174.
3. J. Berstel. *Transductions and Context-Free Languages.* Teubner, Stuttgart, 1979.
4. A. Bertoni, G. Mauri, and N. Sabadini. Membership problems for regular and context-free trace languages. *Information and Computation* 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: C.S. Calude, E. Calude, and M.J. Dinneen (eds.), DLT 2004, Proc., *LNCS 3340*, Springer, Berlin, 2004, 102–113.
6. V. Diekert and G. Rozenberg (eds.). *The Book of Traces*, World Scientific, Singapore, 1995.
7. J. E. Hopcroft and J. D. Ullman. *Introduction to Automata Theory, Languages, and Computation.* Addison-Wesley, Reading, MA, 1979.

8. P. Jančar, F. Moller, and Z. Sawa. Simulation problems for one-counter machines. In: J. Pavelka, G. Tel, and M. Bartošek (eds.), *SOFSEM'99, Proc.*, *LNCS 1725*, Springer, Berlin, 1999, 404–413.
9. M. Kutrib, H. Messerschmidt, and F. Otto. On stateless two-pushdown automata and restarting automata. In: E. Csuhaj-Varjú and Z. Ésik (eds.), *AFL 2008, Proc.*, Hungarian Academy of Sciences, 2008, 257–268.
10. M. Kutrib, H. Messerschmidt, and F. Otto, On stateless two-pushdown automata and restarting automata; *Int. J. Found. Comput. Sci.* 21 (2010) 781–798.
11. H. Messerschmidt and F. Otto. Cooperating distributed systems of restarting automata. *Int. J. Found. Comput. Sci.* 18 (2007) 1333–1342.
12. H. Messerschmidt and F. Otto. On deterministic CD-systems of restarting automata. *Int. J. Found. Comput. Sci.* 20 (2009) 185–209.
13. B. Nagy and F. Otto. CD-systems of stateless deterministic R(1)-automata accept all rational trace languages. In: A.H. Dediu, H. Fernau, and C. Martin-Vide (eds.), *LATA 2010, Proc.*, *LNCS 6031*, Springer, Berlin, 2010, 463–474.
14. B. Nagy and F. Otto. On CD-systems of stateless deterministic R-automata with window size one. Kasseler Informatikschriften, 2/2010. Fachbereich Elektrotechnik/Informatik, Universität Kassel, 2010. https://kobra.bibliothek.uni-kassel.de/handle/urn:nbn:de:hebis:34-2010042732682
15. B. Nagy and F. Otto. Finite-State Acceptors with Translucent Letters. *BILC 2011, Proc.*, to appear.
16. F. Otto. Restarting automata. In: Z. Ésik, C. Martin-Vide, and V. Mitrana (eds.), *Recent Advances in Formal Languages and Applications*, Studies in Computational Intelligence Vol. 25, Springer, Berlin, 2006, 269–303.