Forwards- and Backwards-Reachability for Cooperating Multi-Pushdown Systems

Chris Köcher[1]*[0000−0003−4575−9339] and Dietrich Kuske[2]

[1] Max Planck Institute for Software Systems, Kaiserslautern, Germany
[2] Technische Universität Ilmenau, Germany

Abstract. A cooperating multi-pushdown system consists of a tuple of pushdown systems that can delegate the execution of recursive procedures to sub-tuples; control returns to the calling tuple once all sub-tuples finished their task. This allows the concurrent execution since disjoint sub-tuples can perform their task independently. Because of the concrete form of recursive descent into sub-tuples, the content of the multi-pushdown does not form an arbitrary tuple of words, but can be understood as a Mazurkiewicz trace.
For such systems, we prove that the backwards reachability relation efficiently preserves recognizability, generalizing a result and proof technique by Bouajjani et al. for single-pushdown systems. While this preservation does not hold for the forwards reachability relation, we can show that it efficiently preserves the rationality of a set of configurations; the proof of this latter result is inspired by the work by Finkel et al. It follows that the reachability relation is decidable for cooperating multi-pushdown systems in polynomial time and the same holds, e.g., for safety and liveness properties given by recognizable sets of configurations.

Keywords: Reachability, Formal Verification, Pushdown Automaton, Distributed System

1 Introduction

In this paper, we introduce the model of cooperating multi-pushdown systems[3] and study the reachability relation for such systems. To explain the idea of a cooperating multi-pushdown system, we first look at well-studied pushdown systems. They model the behavior of a sequential recursive program and possess a control state as well as a pushdown. The top symbol of the pushdown stores the execution context, e.g., parameters and local variables, the state can be used to return values from a subroutine to the calling routine. Such a system can, depending on the state and the top symbol, do three types of moves: it can call a subroutine (i.e., change state and top symbol and add a new symbol on top of

* This work was done while Chris Köcher was affiliated with the Technische Universität Ilmenau.

[3] A more descriptive name would be "cooperating systems of pushdown systems", but we refrain from using this term.

the pushdown), it can do an internal action (i.e., change state and top symbol), and it can return from a subroutine (i.e., delete the top symbol and store the necessary information into the state). This leads to the unifying definition of a transition that, depending on state and top symbol, changes state and replaces the top symbol by a (possibly empty) word.

A cooperating multi-pushdown system consists of a finite family of pushdown systems (indexed by a set P). Cooperation is realized by the formation of temporary coalitions that perform a possibly recursive subroutine in a joint manner. Suppose the system is in a configuration where $C \subseteq P$ forms one of the coalitions. The execution context of the joint task is distributed between the top symbols of the pushdowns from the coalition and can only be changed in all these components at once. As above, there are three types of moves depending on the top symbols and the states of the systems from the coalition. First, a (further) subroutine can be called on a sub-coalition $C_0 \subseteq C$. Even more, several subroutines can be called in parallel on disjoint sub-coalitions of C. This is modeled as a change of states and top symbols of C and addition of some further symbols on the pushdowns from subsets of C. Internal actions of the coalition C can change the (common) top symbol as well as the states of the systems that form the coalition C. Similarly, a return move deletes the common top symbol and changes the states of the systems from C, in this moment, the coalition C is dissolved and the systems from C are free to be assigned to new coalitions and tasks by the calling routine. Since several, mutually disjoint coalitions can exist and operate at any particular moment, the cooperating multi-pushdown system is a non-sequential model.

Since a cooperating multi-pushdown system consists of several pushdown systems, a configuration consists of a tuple of local states and a tuple of pushdown contents; the current division into coalitions is modeled by the top symbols of the pushdowns: any component forms a coalition with all components that have the same top symbol a on their stack. Since all these occurrences of the letter a can only change at once, there is some dependency in the tuple of pushdown contents of a configuration. It turns out to be convenient and fruitful to understand such a "consistent" tuple of pushdown contents as a Mazurkiewicz trace. Since the set of all Mazurkiewicz traces forms a monoid, we can define recognizable and rational sets of traces and therefore of configurations: Both these classes of sets of traces enjoy finite representations (by asynchronous automata [19] and NFAs, resp.) that allow to decide membership, any recognizable set is rational but not *vice versa*, any singleton is both, recognizable and rational, and inclusion of a rational set (and therefore in particular of a recognizable set) in a recognizable set is efficiently decidable (but not *vice versa*).

As our main results, we obtain that (1) backwards reachability efficiently preserves the *recognizability* of sets of configurations while (2) forwards reachability efficiently preserves the *rationality*.[4] We also show that asynchronous multi-

[4] The full version of this paper also shows that backwards (forwards) reachability does not preserve rationality (recognizability, resp.)

pushdown systems (a slight generalization of our model) can model 2-pushdown systems and therefore have an undecidable reachability relation.

From our positive results, we infer that the reachability relation as well as certain safety and liveness properties are decidable in polynomial time. Furthermore, the first result implies that EF-model checking is decidable, although one only obtains a non-elementary complexity bound.

Related work. Corresponding results for pushdown systems can be found in [11] and [6] where rationality and recognizability coincide [14]; Finkel et al. gave a simple algorithm proving that the forwards reachability relation preserves the recognizability while this preservation under the backwards reachability relation was shown by Bouajjani et al. Our proof of (1) generalizes the one by Bouajjani et al. while the work by Finkel et al. inspired our proof of (2).

Other forms of multi-pushdown systems have been considered by different groups of authors, e.g., [7, 8, 18, 15, 12, 4, 2, 1, 3, 5, 16]. These alternative models may contain a central control or, similarly to our cooperating systems, local control states. The models can have a fixed number of processes and pushdowns or they are allowed to spawn or terminate other processes. Local processes can differ in their communication mechanism, e.g., by rendevouz or FIFO-channels. The decidability results concern logical formulas of some form or bounded model checking problems.

Mazurkiewicz traces as a form of storage mechanism have been considered by Hutagalung et al. in [13], where multi-buffer systems were studied.

2 Preliminaries

For $R \subseteq S^2$ and $s, t \in S$, let $s\,R := \{t \in S \mid s\,R\,t\}$ and $R\,t := \{s \in S \mid s\,R\,t\}$.

For $n \in \mathbb{N}$, $[n] = \{1, \ldots, n\}$. Let $(S_i)_{i \in [n]}$ be a tuple of sets, $I, J \subseteq [n]$ be two disjoint sets, and $\overline{s} = (s_i)_{i \in [n]}$ and $\overline{t}$ be tuples from $\prod_{i=1}^{n} S_i$. We write $\overline{s}\restriction_I = (s_i)_{i \in I} \in \prod_{i \in I} S_i$ for the restriction of $\overline{s}$ to the components in I and $(\overline{s}\restriction_I, \overline{t}\restriction_J)$ for the joint tuple $\overline{r} \in \prod_{i \in I \cup J} S_i$ with $\overline{r}\restriction_I = \overline{s}\restriction_I$ and $\overline{r}\restriction_J = \overline{t}\restriction_J$.

For a word $w \in A^*$, we write $\mathrm{Alph}(w)$ for the set of letters occurring in w.

A *non-deterministic finite automaton* or *NFA* is a tuple $\mathfrak{A} = (Q, A, I, \delta, F)$ where Q is a finite set of *states*, A is an alphabet, $I, F \subseteq Q$ are sets of *initial* and *accepting* states, respectively, and $\delta \subseteq Q \times A \times Q$ is a set of *transitions*; its size $\|\mathfrak{A}\|$ is $|Q| + |A|$. We write $Q_1 \xrightarrow{w}_{\mathfrak{A}} Q_2$ if there is a run from some state $p \in Q_1$ to some state $q \in Q_2$ labeled with w in $\mathfrak{A}$; $\{p\} \xrightarrow{w}_{\mathfrak{A}} \{q\}$ is abbreviated $p \xrightarrow{w}_{\mathfrak{A}} q$. The *language accepted by* $\mathfrak{A}$ is $L(\mathfrak{A}) := \{w \in A^* \mid I \xrightarrow{w}_{\mathfrak{A}} F\}$.

We will model the contents of our multi-pushdown systems with the help of Mazurkiewicz traces; for a comprehensive survey of this topic we refer to [10]. Traces were first studied in [9] as "heaps of pieces" and later introduced into computer science by Mazurkiewicz to model the behavior of a distributed system [17]. The fundamental idea is that any letter $a \in A$ is assigned a set of *locations* or *processes* $a\,\mathcal{L} \subseteq P$ it operates on (where P is some set):

A *distributed alphabet* is a triple $\mathcal{D} = (A, P, \mathcal{L})$ where A and P are two alphabets of *letters* and *processes*, respectively, and $\mathcal{L} \subseteq A \times P$ associates letters to processes such that $a\,\mathcal{L} \neq \emptyset$ for each $a \in A$. In this paper, $\mathcal{D}$ will always denote a distributed alphabet $(A, P, \mathcal{L})$.

For a word $w \in A^*$ we denote the set of processes associated with w by $w\,\mathcal{L} := \bigcup_{a \in \mathrm{Alph}(w)} a\,\mathcal{L} \subseteq P$. In particular, we set $\varepsilon\,\mathcal{L} := \emptyset$. By $\pi_i \colon A^* \to A_i^*$ we denote the *projection* onto $A_i := \mathcal{L}\,i$ (the alphabet of all letters associated to process i), i.e., the monoid morphism with $\pi_i(a) = a$ for $a \in A_i$ and $\pi_i(b) = \varepsilon$ for $b \in A \setminus A_i$.

Since $\pi_i \colon A^* \to A_i^*$ is a monoid morphism for all $i \in [n]$, also the mapping

$$\overline{\pi} \colon A^* \to \prod_{i \in P} A_i^* \colon w \mapsto (\pi_i(w))_{i \in P}$$

is a monoid morphism. For $w \in A^*$, we call $\overline{\pi}(w)$ the *(Mazurkiewicz) trace* induced by w. The *trace monoid* is the submonoid of $\prod_{i \in P} A_i^*$ with universe $\mathbb{M}(\mathcal{D}) = \{\overline{\pi}(w) \mid w \in A^*\}$; its elements are *traces* and its subsets are *trace languages.*

We call two words $v, w \in A^*$ with $v\,\mathcal{L} \cap w\,\mathcal{L} = \emptyset$ *independent* and denote this fact by $v \parallel w$. We can see that $v \parallel w$ implies $\overline{\pi}(vw) = \overline{\pi}(wv)$.

Let $\mathfrak{A} = (Q, A, I, \delta, F)$ be an NFA. The *accepted trace language* of $\mathfrak{A}$ is $T(\mathfrak{A}) := \{\overline{\pi}(w) \mid I \xrightarrow{w}_{\mathfrak{A}} F\}$. In other words, $T(\mathfrak{A})$ is the image of $L(\mathfrak{A})$ under the morphism $\overline{\pi}$. A trace language $L \subseteq \mathbb{M}(\mathcal{D})$ is called *rational* if there is an NFA $\mathfrak{A}$ with $T(\mathfrak{A}) = L$, i.e., iff L is the image of some regular language in A^* under the morphism $\overline{\pi}$. A trace language L is *recognizable* iff its preimage under the morphism $\overline{\pi}$, i.e. $\{w \in A^* \mid \overline{\pi}(w) \in L\}$, is regular. Clearly, any recognizable trace language is rational. The converse implication holds only in case any two letters are dependent.

A finite automaton that reads letters of a distributed alphabet should consist of components for all $i \in P$ such that any letter $a \in A$ acts only on the components from $a\,\mathcal{L}$. This idea leads to the following definition of an asynchronous automaton. But first, we fix a particular notation: For a tuple $(Q_i)_{i \in P}$ of finite sets Q_i, we write $\mathbf{Q}$ for the direct product $\prod_{i \in P} Q_i$.

Definition 2.1. *An* asynchronous automaton *or* AA *is an NFA* $\mathfrak{A} = (Q, A, I, \delta, F)$ *where* $Q = \mathbf{Q}$ *is the product of finite sets* Q_i *of* local states *and where, for every* $(\overline{p}, a, \overline{q}) \in \delta$ *and* $\overline{r} \in \prod_{i \in P \setminus a\,\mathcal{L}} Q_i$*, we have*

$$(i)\ \overline{p}{\restriction_{P \setminus a\,\mathcal{L}}} = \overline{q}{\restriction_{P \setminus a\,\mathcal{L}}} \text{ and } (ii)\ ((\overline{p}{\restriction_{a\,\mathcal{L}}}, \overline{r}), a, (\overline{q}{\restriction_{a\,\mathcal{L}}}, \overline{r})) \in \delta\,.$$

Here, (i) ensures that any a-transition of $\mathfrak{A}$ only modifies components from $a\,\mathcal{L}$ while the other components are left untouched, and (ii) guarantees that a-transitions are insensitive to the local states of the components in $P \setminus a\,\mathcal{L}$.

Every asynchronous automaton accepts a recognizable trace language. Conversely, every recognizable trace language $L \subseteq \mathbb{M}(\mathcal{D})$ is accepted by some deterministic asynchronous automaton [19].

3 Introducing Cooperating Multi-Pushdown Systems

An AA consists of several NFAs that synchronize by joint actions. In a similar manner, we will now consider several pushdown systems.

Recall that a pushdown system (or PDS) consists of a control unit (that can be in any of finitely many control states) and a pushdown (that can hold words over the pushdown alphabet A). Its transitions read the top letter a from the pushdown, write a word w onto it, and change the control state. In our model, we have a pushdown system $\mathfrak{P}_i$ for every $i \in P$ whose pushdown alphabet is A_i. These systems synchronize by the letters read and written onto their pushdown.

Definition 3.1. *An* asynchronous multi-pushdown system *or* aPDS *is a tuple* $\mathfrak{P} = (Q, \Delta)$ *where* $Q = \mathbf{Q}$ *holds for some finite sets* Q_i *of* local states *and* $\Delta \subseteq Q \times A \times A^* \times Q$ *is a finite set of* transitions *such that, for each transition* $(\overline{p}, a, w, \overline{q}) \in \Delta$ *and* $\overline{r} \in \prod_{i \in P \setminus aw\,\mathcal{L}} Q_i$*, we have*

$$(i)\ \overline{p}{\restriction}_{P \setminus aw\,\mathcal{L}} = \overline{q}{\restriction}_{P \setminus aw\,\mathcal{L}} \text{ and } (ii)\ ((\overline{p}{\restriction}_{aw\,\mathcal{L}}, \overline{r}), a, w, (q{\restriction}_{aw\,\mathcal{L}}, \overline{r})) \in \Delta\,.$$

Its size $\|\mathfrak{P}\|$ *is* $|Q| + k \cdot |\Delta|$ *where* $k-1$ *is the maximal length of a word written by any of the transitions (i.e.,* $\Delta \subseteq Q \times A \times A^{<k} \times Q$*).*

The set of configurations $\mathrm{Conf}_{\mathfrak{P}}$ *of* $\mathfrak{P}$ *equals* $Q \times \mathbb{M}(\mathcal{D})$*. For two configurations* $(\overline{p}, \overline{\pi}(u)), (\overline{q}, \overline{\pi}(v)) \in \mathrm{Conf}_{\mathfrak{P}}$ *we set* $(\overline{p}, \overline{\pi}(u)) \vdash (\overline{q}, \overline{\pi}(v))$ *if there is a transition* $(\overline{p}, a, w, \overline{q}) \in \Delta$ *and a word* $x \in A^*$ *with* $\overline{\pi}(u) = \overline{\pi}(ax)$ *and* $\overline{\pi}(v) = \overline{\pi}(wx)$*. The reflexive and transitive closure of* $\vdash$ *is the reachability relation* $\vdash^*$*.*

Let C *and* D *be sets of configurations.*

- *We write* $C \vdash^* D$ *if there are* $c \in C$ *and* $d \in D$ *with* $c \vdash^* d$*.*
- *The set* C *is* rational *(*recognizable*, resp.) if, for all* $\overline{q} \in Q$*, the trace language* $C_{\overline{q}} := \{\overline{\pi}(u) \mid (\overline{q}, \overline{\pi}(u)) \in C\}$ *is rational (recognizable, resp.).*
- $\mathrm{pre}_{\mathfrak{P}}(C) := \{c \in \mathrm{Conf}_{\mathfrak{P}} \mid c \vdash C\}$ *and* $\mathrm{post}_{\mathfrak{P}}(C) := \{d \in \mathrm{Conf}_{\mathfrak{P}} \mid C \vdash d\}$ *are the sets of predecessors/successors of configurations from* C*, and*

$$\mathrm{pre}^*_{\mathfrak{P}}(C) := \bigcup_{k \in \mathbb{N}} \mathrm{pre}^k_{\mathfrak{P}}(C) \text{ and } \mathrm{post}^*_{\mathfrak{P}}(C) := \bigcup_{k \in \mathbb{N}} \mathrm{post}^k_{\mathfrak{P}}(C)$$

  are the sets of configurations backwards *(*forwards*, resp.) reachable from some configuration in* C*.*

The reachability relation for configurations of asynchronous multi-pushdown systems is, in general, undecidable:

Theorem 3.2. *There exists an aPDS with undecidable reachability relation* $\vdash^*$*.*

Proof. We start with a classical 2-pushdown system $\mathfrak{P}$ with an undecidable reachability relation (its set of states is Q and the two pushdowns use disjoint alphabets A_1 and A_2). Let $A = A_1 \cup A_2 \cup \{\top\}$ and $P = [2]$. We consider the distributed alphabet $\mathcal{D}$ with $a\,\mathcal{L} = \{i\}$ for $a \in A_i$ and $\top\,\mathcal{L} = \{1, 2\}$.

We simulate $\mathfrak{P}$ by an aPDS $\mathfrak{P}'$ over $\mathcal{D}$ as follows. The first process of $\mathfrak{P}'$ stores the state of the simulated system $\mathfrak{P}$ together with a letter from A_1 or

ε, i.e., $Q_1 = Q(A_1 \cup \{\varepsilon\})$, the second process can store a letter from A_2 or the empty word, i.e., $Q_2 = A_2 \cup \{\varepsilon\}$.

A transition $(p,(a,b),(u,v),q)$ of $\mathfrak{P}$ (that replaces a and b by u and v on the two pushdowns) is simulated by three transitions of the aPDS: $((p\varepsilon,.),a,\varepsilon,(pa,.))$ reads a from the first pushdown and stores it in the first local state; $((.,\varepsilon),b,\top,(.,b))$ reads b from the second pushdown, stores it in the second local state, and puts $\top$ onto both pushdowns; finally, $((pa,b),\top,uv,(q\varepsilon,\varepsilon))$ replaces $\top$ by uv (i.e., $\pi_1(uv) = u$ is written onto the first pushdown and $\pi_2(uv) = v$ onto the second). $\square$

To obtain a model with a decidable reachability relation, we therefore have to restrict aPDS.[5] To this aim, we require that any transition can only write onto pushdowns it reads from.

Definition 3.3. *A* cooperating multi-pushdown system *or* cPDS *is an aPDS $\mathfrak{P} = (\mathbf{Q},\Delta)$ with $w\,\mathcal{L} \subseteq a\,\mathcal{L}$ for each transition $(\overline{p},a,w,\overline{q}) \in \Delta$.*

Example 3.4. Suppose $\mathcal{D} = (\{a,b,c\},\{1,2\},\{(a,1),(a,2),(b,1),(c,2)\})$. We consider the cPDS $\mathfrak{P}$ from Figure 1 where edges from global state $\overline{p}$ to global state $\overline{q}$ labeled $a \mid w$ visualize global transitions $(\overline{p},a,w,\overline{q})$. The set of global states of $\mathfrak{P}$ is the product $\{p_1,q_1\} \times \{p_2,q_2\}$. Additionally, the transitions reading b and c only depend on process 1 and 2, resp. Since $b\,\mathcal{L}, c\,\mathcal{L} \subseteq a\,\mathcal{L}$, any global transition $(\overline{p},x,w,\overline{q})$ satisfies $w\,\mathcal{L} \subseteq x\,\mathcal{L}$, i.e., $\mathfrak{P}$ is, indeed, a cPDS.

The following sequence is a run of $\mathfrak{P}$ from $((p_1,p_2),\overline{\pi}(ac))$ to $((q_1,q_2),\overline{\pi}(bb))$:

$$\begin{aligned}((p_1,p_2),\overline{\pi}(ac)) &\vdash ((q_1,p_2),\overline{\pi}(abc)) \vdash ((q_1,p_2),\overline{\pi}(abbc))\\ &\vdash ((q_1,q_2),\overline{\pi}(bbc)) \vdash ((q_1,q_2),\overline{\pi}(bb))\,.\end{aligned}$$

In order to decide the reachability relation, we will compute, from a set of configurations C, the set $\mathrm{pre}^*_{\mathfrak{P}}(C)$. To represent possibly infinite sets of configurations, we use $\mathfrak{P}$-asynchronous automata (defined next).

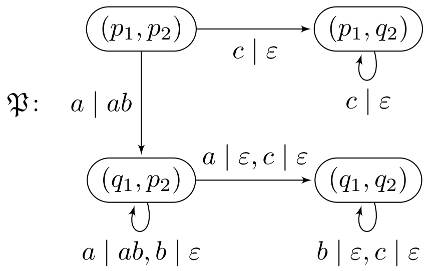
$\mathfrak{P}$:
(p_1,p_2)
(p_1,q_2)
(q_1,p_2)
(q_1,q_2)
$c \mid \varepsilon$
$c \mid \varepsilon$
$a \mid ab$
$a \mid \varepsilon, c \mid \varepsilon$
$a \mid ab, b \mid \varepsilon$
$b \mid \varepsilon, c \mid \varepsilon$

Fig. 1. The cPDS $\mathfrak{P}$ from Example 3.4.

Definition 3.5. *Let $\mathfrak{P} = (\mathbf{Q},\Delta)$ be a cPDS. A* $\mathfrak{P}$-asynchronous automaton *or* $\mathfrak{P}$-AA *is an AA $\mathfrak{A} = (\mathbf{S},A,\emptyset,\delta,F)$ such that $Q_i \subseteq S_i$ for all $i \in P$.*

The $\mathfrak{P}$-AA $\mathfrak{A}$ accepts *the following set $C(\mathfrak{A})$ of configurations of $\mathfrak{P}$:*

$$\{(\overline{q},\overline{\pi}(w)) \in \mathrm{Conf}_{\mathfrak{P}} \mid \overline{q} \in \mathbf{Q}, \overline{q} \xrightarrow{w}_{\mathfrak{A}} F\}\,.$$

[5] The proof of Theorem 3.2 shows that requiring aw to be connected for any transition $(\overline{p},a,w,\overline{q})$ does not yield decidability.

By the very definition, any set $C(\mathfrak{A})$ is a recognizable set of configurations. Conversely, suppose $C \subseteq \mathrm{Conf}_{\mathfrak{P}}$ is recognizable such that, by definition, all the languages $C_{\overline{q}}$ are recognizable. Then we can represent each of the languages $C_{\overline{q}}$ by an AA $\mathfrak{A}_{\overline{q}}$. Since $\overline{q}$ is a P-tuple, we can assume, without loss of generality, that $\overline{q}$ is the only initial state of the AA $\mathfrak{A}_{\overline{q}}$. Following Bouajjani et al. [6], we can further assume that all these AAs differ in their initial state, only. Thus, we obtain the following result.

Observation 3.6. *Let $\mathfrak{P} = (\mathbf{Q}, \Delta)$ be a cPDS. A set of configurations $C \subseteq \mathrm{Conf}_{\mathfrak{P}}$ is* recognizable *if, and only if, there is a $\mathfrak{P}$-AA $\mathfrak{A}$ with $C(\mathfrak{A}) = C$.*

4 Computing the Backwards Reachable Configurations

In this section we want to compute the backwards reachable configurations in a cPDS $\mathfrak{P}$. The main result of this section states that the mapping $\mathrm{pre}^*_{\mathfrak{P}}$ effectively preserves the recognizability of sets of configurations.

Theorem 4.1. *Let $\mathfrak{P} = (\mathbf{Q}, \Delta)$ be a cPDS and $C \subseteq \mathrm{Conf}_{\mathfrak{P}}$ be a recognizable set of configurations. Then the set $\mathrm{pre}^*_{\mathfrak{P}}(C)$ is recognizable.*

Even more, from $\mathcal{D}$, $\mathfrak{P}$, and a $\mathfrak{P}$-AA $\mathfrak{A}^{(0)}$, one can construct in polynomial time a $\mathfrak{P}$-AA $\mathfrak{A}$ that accepts the set $\mathrm{pre}^*_{\mathfrak{P}}(C(\mathfrak{A}^{(0)}))$.

The rest of this section is devoted to the proof of this result.

Adapting ideas by Bouajjani et al. [6] from NFAs to AA, we construct a $\mathfrak{P}$-AA $\mathfrak{A}$ that accepts the set $\mathrm{pre}^*_{\mathfrak{P}}(C(\mathfrak{A}^{(0)}))$ of configurations backwards reachable from $C(\mathfrak{A}^{(0)})$. To this aim, we will inductively add new transitions to the $\mathfrak{P}$-AA $\mathfrak{A}^{(0)} = (\mathbf{S}, A, \emptyset, \delta^{(0)}, F)$, but leave the sets of states, initial states, and accepting states unchanged. We can assume (and this assumption is crucial for the correctness of the construction) that the automaton cannot enter a local state from the cPDS $\mathfrak{P}$, i.e., we have $q_i \in S_i \setminus Q_i$ for any $(\overline{p}, a, \overline{q}) \in \Delta$ and any $i \in a\mathcal{L}$.

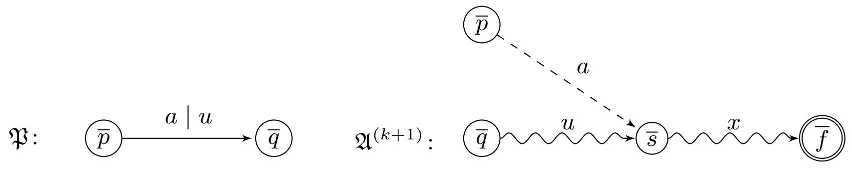

Fig. 2. Visualization of the construction of $\mathfrak{A}^{(k+1)}$.

Suppose that we already constructed the $\mathfrak{P}$-AA $\mathfrak{A}^{(k)}$. To obtain $\mathfrak{A}^{(k+1)}$ from $\mathfrak{A}^{(k)}$, we just add all transitions $(\overline{p}, a, \overline{s})$ with $(\overline{p}, a, u, \overline{q}) \in \Delta$ and $\overline{q} \xrightarrow{u}_{\mathfrak{A}^{(k)}} \overline{s}$ for some $\overline{q} \in \mathbf{Q}$ and $u \in A^*$ (see Fig. 2). Note that $(\overline{p}, a, u, \overline{q}) \in \Delta$ as well as the run $\overline{q} \xrightarrow{u}_{\mathfrak{A}^{(k)}} \overline{s}$ operate on components from $a\,\mathcal{L} \supseteq u\,\mathcal{L}$, only. Hence, the same applies to the new transition $(\overline{p}, a, \overline{s})$ ensuring that $\mathfrak{A}^{(k+1)}$ is asynchronous (this argument requires $a\,\mathcal{L} \supseteq u\,\mathcal{L}$ and would therefore not work for aPDS).

The "limit" of this construction is the $\mathfrak{P}$-AA $\mathfrak{A}^{(\infty)} = (\mathbf{S}, A, \emptyset, \delta^{(\infty)}, F)$ with $\delta^{(\infty)} = \bigcup_{k\in\mathbb{N}} \delta^{(k)}$.

Example 4.2. Recall the cPDS $\mathfrak{P}$ from Example 3.4. In Fig. 3 we depict our algorithm on input $\mathfrak{P}$ and the set of configurations $C = \{((q_1, q_2), \varepsilon)\}$. A $\mathfrak{P}$-AA $\mathfrak{A}^{(0)} = (S_1 \times S_2, A, \emptyset, \delta, F)$ accepting this set is depicted in the left.

In $\mathfrak{A}^{(1)}$, we have $(q_1, p_2) \xrightarrow{ab}_{\mathfrak{A}^{(1)}} (q_1, q_2)$ (depicted in bold and red) and, in $\mathfrak{P}$, we have the transition $((p_1, p_2), a, ab, (q_1, p_2)) \in \Delta$. The definition of $\delta^{(2)}$ implies that $((p_1, p_2), a, (q_1, q_2))$ is a new local transition.

The construction terminates with $\mathfrak{A}^{(2)}$ which is a $\mathfrak{P}$-AA that accepts the set of configurations backwards reachable from $C = \big\{((q_1, q_2), \varepsilon)\big\}$.

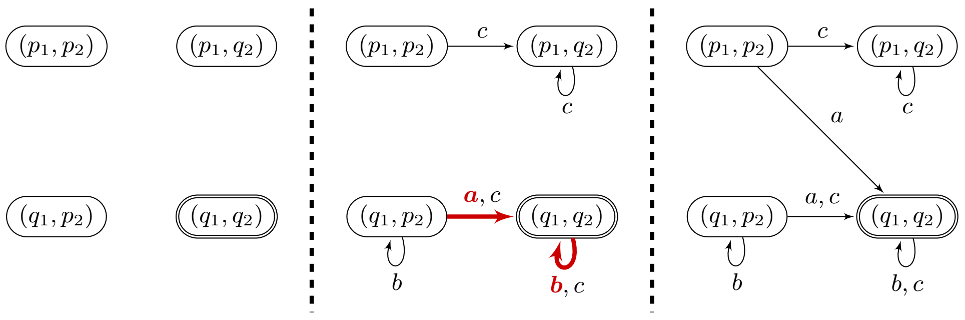

Fig. 3. The $\mathfrak{P}$-AA $\mathfrak{A}^{(0)}$, $\mathfrak{A}^{(1)}$, and $\mathfrak{A}^{(2)}$ (from left to right) from Example 4.2.

Now, we show $C(\mathfrak{A}^{(\infty)}) = \mathrm{pre}^*_{\mathfrak{P}}(C(\mathfrak{A}^{(0)}))$. First, by induction on $k \in \mathbb{N}$, one can easily prove $\mathrm{pre}^k_{\mathfrak{P}}(C(\mathfrak{A}^{(0)})) \subseteq C(\mathfrak{A}^{(k)})$ (which ensures the inclusion "$\supseteq$"). On the other hand, Example 4.2 shows that the converse inclusion $C(\mathfrak{A}^{(k)}) \subseteq \mathrm{pre}^k_{\mathfrak{P}}(C(\mathfrak{A}^{(0)}))$ does not necessarily hold. The following lemma is the central argument in the proof of the inclusion "$\subseteq$" as it allows to infer $C(\mathfrak{A}^{(k)}) \subseteq \mathrm{pre}^*_{\mathfrak{P}}(C(\mathfrak{A}^{(0)}))$ for all $k \in \mathbb{N}$.

Lemma 4.3. *Let $k \in \mathbb{N}$, $v \in A^*$, $\overline{p} \in \mathbf{Q}$, and $\overline{s} \in \mathbf{S}$ with $\overline{p} \xrightarrow{v}_{\mathfrak{A}^{(k)}} \overline{s}$. Then there are a global state $\overline{r} \in \mathbf{Q}$ and a word $w \in A^*$ with the following properties:*

$$\text{(a) } (\overline{p}, \overline{\pi}(v)) \vdash^* (\overline{r}, \overline{\pi}(w)) \text{ and (b) } \overline{r} \xrightarrow{w}_{\mathfrak{A}^{(0)}} \overline{s}.$$

Proof idea. We only indicate where our proof differs significantly from a similar one from [6] for pushdown systems. In general, the lemma is shown by induction on k, and the significant difference occurs in the induction step. So assume the lemma holds for k. To prove it for $k+1$, one proceeds by induction on the length of the word v. Again, the significant difference occurs in the induction step. Hence, we assume that the lemma holds for $k+1$ and any word of length at most n and we will prove it for a word $v = v'a \in A^{n+1}$ with $a \in A$ and $v' \in A^n$.

So let $\overline{p} \in \mathbf{Q}$ and $\overline{s} \in \mathbf{S}$ such that $\overline{p} \xrightarrow{v}_{\mathfrak{A}^{(k+1)}} \overline{s}$. Since $\overline{p} \xrightarrow{v'a}_{\mathfrak{A}^{(k+1)}} \overline{s}$, there is some global state $\overline{s'} \in \mathbf{S}$ with $\overline{p} \xrightarrow{v'}_{\mathfrak{A}^{(k+1)}} \overline{s'} \xrightarrow{a}_{\mathfrak{A}^{(k+1)}} \overline{s}$.

Since $|v'| = n$, the inductive hypothesis provides a global state $\overline{q'} \in \mathbf{Q}$ and a word $w' \in A^*$ with $(\overline{p}, \overline{\pi}(v')) \vdash^* (\overline{q'}, \overline{\pi}(w'))$ and $\overline{q'} \xrightarrow{w'}_{\mathfrak{A}^{(0)}} \overline{s'}$. Note that the former implies in particular $(\overline{p}, \overline{\pi}(v)) = (\overline{p}, \overline{\pi}(v'a)) \vdash^* (\overline{q'}, \overline{\pi}(w'a))$. The difficult case is if $\overline{s'} \xrightarrow{a}_{\mathfrak{A}^{(k)}} \overline{s}$ does not hold (see Figure 4 where edges $\xrightarrow{w}_{\mathfrak{A}^{(\ell)}}$ are denoted by $\xrightarrow{w}_{\ell}$). Then $(\overline{s'}, a, \overline{s})$ is a new transition in $\delta^{(k+1)}$ which implies $s'_i \in Q_i$ for all $i \in a\,\mathcal{L}$. Recall that the $\mathfrak{P}$-AA $\mathfrak{A}^{(0)}$ cannot enter a local state of the pushdown system. Here, our argument differs from the one in [6]), as we can only infer $w'\,\mathcal{L} \cap a\,\mathcal{L} = \emptyset$. Setting $\overline{t} = (\overline{s}{\restriction_{a\,\mathcal{L}}}, \overline{q'}{\restriction_{w'\,\mathcal{L}}}, \overline{s'}{\restriction_{P \setminus w'a\,\mathcal{L}}})$, one can nevertheless complete the picture. $\square$

$$\begin{tikzcd}
\overline{p} \arrow[r, "v'"] \arrow[r, "k+1"'] & \overline{s'} \arrow[r, "a"] \arrow[r, "k+1"'] & \overline{s} \\
\overline{q'} \arrow[ur, "w'"] \arrow[ur, "0"'] \arrow[r, "a"] \arrow[r, "k+1"'] & \overline{t} \arrow[ur, "w'"] \arrow[ur, "0"'] & \\
\overline{r} \arrow[ur, "w''"] \arrow[ur, "0"'] & &
\end{tikzcd}$$

Fig. 4. Visualization of the proof idea

The remaining arguments for Theorem 4.1 are those from [6].

5 Computing the Forwards Reachable Configurations

The main result of this section is that the mapping $\mathrm{post}^*_{\mathfrak{P}}$ efficiently preserves rationality. Here, we represent a rational set of configurations $C \subseteq \mathbf{Q} \times \mathbb{M}(\mathcal{D})$ by a tuple $\overline{\mathfrak{A}}$ of NFAs $\mathfrak{A}_{\overline{q}}$ for $\overline{q} \in \mathbf{Q}$ that, for all global states $\overline{q}$, accept the trace language $T(\mathfrak{A}_{\overline{q}}) = C_{\overline{q}} = \{\overline{\pi}(w) \mid (\overline{q}, \overline{\pi}(w)) \in C\}$. If this is the case, we say "the tuple $\overline{\mathfrak{A}}$ accepts C".

Theorem 5.1. *Let $\mathfrak{P}$ be a cPDS and $C \subseteq \mathrm{Conf}_{\mathfrak{P}}$ be rational. Then $\mathrm{post}^*_{\mathfrak{P}}(C)$ is rational. In particular, we can compute a tuple of NFAs accepting $\mathrm{post}^*_{\mathfrak{P}}(C)$ from $\mathfrak{P}$ and a tuple of NFAs accepting C. If $\mathcal{D}$ is fixed, this construction is possible in polynomial time.*

The proof of this theorem is inspired by the work by Finkel et al. [11]. To explain its idea and particularities, we first start with a classical pushdown system $\mathfrak{P} = (Q_1, \Delta)$. Suppose there are transitions (p, a, bv, q) and (q, b, ε, r) implying $(p, ax) \vdash (q, bvx) \vdash (r, vx)$ for any word x. If we add the transition (p, a, v, r) to Δ that allows to go from (p, ax) to (r, vx) in one step, the reachability relation does not change. We keep adding such "shortcuts" and call the resulting pushdown system $\mathfrak{P}^{(\infty)}$. Then, any run of the original system $\mathfrak{P}$ can be simulated by a run of the system with shortcuts $\mathfrak{P}^{(\infty)}$ that first shortens the pushdown and then writes onto the pushdown. It follows that, for pushdown systems $\mathfrak{P}$, the mapping $\mathrm{post}^*_{\mathfrak{P}}$ preserves rationality.

The crucial point of the above construction is that any run of the system $\mathfrak{P}^{(\infty)}$ can be brought into some "simple form" by using shortcuts. Here, "simple form" means that it consists of two phases: the pushdown decreases properly in every step of the first phase and does not decrease in any step of the second phase.

Our strategy in the proof of Theorem 5.1 will extend the above idea:

1. First, one demonstrates that Theorem 5.1 holds for "homogeneous" systems that formalize and strengthen the two types of phases from above:
   A cPDS $\mathfrak{P} = (\mathbf{Q}, \Delta)$ is *homogeneous* if one of the following holds.
   (1) All transitions $(\overline{p}, a, w, \overline{q}) \in \Delta$ satisfy $w = \varepsilon$.
   (2) There is $X \subseteq P$ such that all transitions $(\overline{p}, a, w, \overline{q}) \in \Delta$ satisfy $a\,\mathcal{L} = X$ and $w \neq \varepsilon$.
   This means, $\mathfrak{P}$ is homogeneous if either no transition writes anything or if all transitions read exactly from the same subset $X \subseteq P$ of processes and write at least one letter. In particular, in the second case we have $a\,\mathcal{L} = b\,\mathcal{L}$ for each pair of transitions $(\overline{p}, a, v, \overline{q}), (\overline{r}, b, w, \overline{s}) \in \Delta$ (but not necessarily $a = b$).
2. Using the result on homogeneous systems, one demonstrates Theorem 5.1 for "saturated" systems, i.e., systems where no new "shortcuts" can be added:
   A cPDS $\mathfrak{P} = (\mathbf{Q}, \Delta)$ is *saturated* if $(\overline{p}, a, ubv, \overline{q}), (\overline{q}, b, \varepsilon, \overline{r}) \in \Delta$ with $u \parallel b$ implies $(\overline{p}, a, uv, \overline{r}) \in \Delta$.
   This step differs significantly from the above arguments from [11] as the main difficulty is to show that the number of "phases" can be bounded (the bound is linear in the number of sets $a\,\mathcal{L} \subseteq P$ which is bounded by $|A|$).
3. Finally, Proposition 5.6 proves Theorem 5.1 in full generality by showing that any system can be saturated by adding shortcuts.

5.1 Forwards Reachability in Homogeneous Systems

Let $\mathfrak{P} = (\mathbf{Q}, \Delta)$ be a cPDS and let D be a rational set of configurations.

If $\Delta \subseteq \mathbf{Q} \times A \times \{\varepsilon\} \times \mathbf{Q}$, then any transition shortens the pushdowns. Hence, the effect of $\mathrm{post}^*_{\mathfrak{P}}$ is a left quotient of $D_{\overline{q}}$ wrt. a *recognizable* trace language.

Now suppose $X \subseteq P$ and $a\,\mathcal{L} = X$ as well as $u \neq \varepsilon$ for all transitions $(\overline{p}, a, u, \overline{q}) \in \Delta$. Then, dually to the above case, the effect of $\mathrm{post}^*_{\mathfrak{P}}$ is the concatenation of $D_{\overline{q}}$ and a rational (not necessarily recognizable) trace language.

It follows (in both cases), that $\mathrm{post}^*_{\mathfrak{P}}(D)$ is rational. A closer analysis reveals that also the remaining claims of Theorem 5.1 hold for homogeneous systems.

5.2 Forwards Reachability in Saturated Systems

Recall the constructed pushdown system with just one pushdown $\mathfrak{P}^{(\infty)}$ from the beginning of this section. This pushdown system is saturated. We learned that any run in $\mathfrak{P}^{(\infty)}$ can be simulated by a run consisting of two phases: first, the pushdown shortens and then, it increases. The following example shows that this is not possible in systems with more than one pushdown.

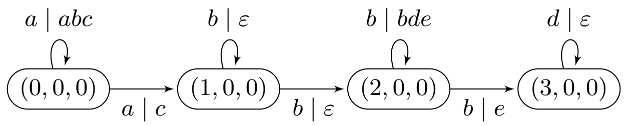

Fig. 5. The cPDS from Example 5.2.

Example 5.2. We consider the distributed alphabet $\mathcal{D} = (A, P, \mathcal{L})$ with $A = \{a, b, c, d, e\}$ and $P = \{1, 2, 3\}$ where $a\,\mathcal{L} = P$, $b\,\mathcal{L} = \{1, 2\}$, $c\,\mathcal{L} = \{3\}$, $d\,\mathcal{L} = \{1\}$, and $e\,\mathcal{L} = \{2\}$. Further, let $\mathfrak{P} = (\mathbf{Q}, \Delta)$ be the saturated cPDS from Figure 5. The following is the only run from the configuration $(\overline{0}, \overline{\pi}(a))$ to the configuration $(\overline{3}, \overline{\pi}(e^4c^4))$ where we write $\overline{n}$ for the state $(n, 0, 0)$:

$$\begin{aligned}(\overline{0}, \overline{\pi}(a)) &\vdash^3 (\overline{0}, \overline{\pi}(abcbcbc)) \vdash (\overline{1}, \overline{\pi}(cbcbcbc)) = (\overline{1}, \overline{\pi}(b^3c^4))\\ &\vdash^2 (\overline{2}, \overline{\pi}(bc^4))\\ &\vdash^3 (\overline{2}, \overline{\pi}(bdededec^4)) \vdash (\overline{3}, \overline{\pi}(edededec^4)) = (\overline{3}, \overline{\pi}(d^3e^4c^4))\\ &\vdash^3 (\overline{3}, \overline{\pi}(e^4c^4))\end{aligned}$$

Note that this run splits into four phases (that correspond to the four lines above); it increases its pushdowns in the first and third and decreases them in the second and fourth.

So far, we used the term “phase” without defining it formally. To be a bit more precise, a “phase” is a run of some maximal homogeneous subsystem of $\mathfrak{P}$. These subsystems are defined next.

Definition 5.3. *Let $\mathfrak{P} = (\mathbf{Q}, \Delta)$ be a cPDS.*

1. *Let $\Delta_\varepsilon = \{(\overline{p}, a, \varepsilon, \overline{q}) \in \Delta\}$ and $\mathfrak{P}_\varepsilon = (\mathbf{Q}, \Delta_\varepsilon)$.*
2. *For $X \subseteq P$, let $\Delta_X = \{(\overline{p}, a, u, \overline{q}) \in \Delta \mid a\,\mathcal{L} = X \text{ and } u \neq \varepsilon\}$ and $\mathfrak{P}_X = (\mathbf{Q}, \Delta_X)$.*

To simplify notation, we write $\vdash_\varepsilon$ for $\vdash_{\mathfrak{P}_\varepsilon}$ and $\vdash_X$ for $\vdash_{\mathfrak{P}_X}$ for any $X \subseteq P$.

Since Δ is the disjoint union of the subsets Δ_ε and Δ_X for $X \subseteq P$, any run of $\mathfrak{P}$ splits uniquely into maximal subruns of these subsystems and these subruns are precisely what we called “phase”.

For $X \subseteq P$, set $\Vdash_X = \vdash_\varepsilon^* \circ \vdash_X^+ \subseteq \mathrm{Conf}_{\mathfrak{P}} \times \mathrm{Conf}_{\mathfrak{P}}$. In other words, $c_1 \Vdash_X c_2$ means that the system $\mathfrak{P}$ has a run from c_1 to c_2 that first shortens the pushdowns and then, in the second phase, uses transitions from Δ_X, only. Note that the first (deleting) phase is allowed to be empty while the second (writing) phase is required to be non-empty.

For $\overline{X} = (X_i)_{i\in[n]}$ with $X_i \subseteq P$, set $\Vdash_{\overline{X}} = \Vdash_{X_1} \circ \Vdash_{X_2} \circ \cdots \circ \Vdash_{X_n}$.

The binary relation $\vdash^*$ is the union of all relations $\Vdash_{\overline{X}} \circ \vdash_\varepsilon^*$ for $\overline{X}$ a sequence of subsets of P of arbitrary length. Our next aim is to show that we only need to consider sequences $\overline{X}$ of bounded length. The central lemma proves that, under certain conditions, $\Vdash_{X_0} \circ \Vdash_{\overline{X}}$ is contained in $\Vdash_{\overline{X}}$, i.e., that we can shorten the sequence $X_0\overline{X}$.

Lemma 5.4. *Let $\mathfrak{P} = (\mathbf{Q}, \Delta)$ be a saturated cPDS. Let $X_0, X_1, \ldots, X_{n+1} \subseteq P$ such that* (i) *$X_0 \subseteq X_{n+1}$ and* (ii) *$X_i \not\subseteq X_{n+1}$ for all $1 \le i \le n$.*

Then $\Vdash_{(X_0,X_1,\ldots,X_{n+1})} \subseteq \Vdash_{(X_1,\ldots,X_{n+1})}$.

Proof idea. The central argument of the proof goes as follows: Suppose we have

$$c_0 \vdash c_1 \Vdash_{(X_1,X_2,\ldots,X_n)} \circ \vdash_\varepsilon^* \circ \vdash_{X_{n+1}} d$$

and let $(\overline{p}, a, u, \overline{q}) \in \Delta$ with $a\,\mathcal{L} = X_0$ denote the transition used in the first step. The proof then proceeds by induction on the length of the word u. If $u = \varepsilon$, we get $c_0 \vdash_\varepsilon c_1$ implying $c_0 \Vdash_{(X_1,\ldots,X_{n+1})} d$. Now let $u \neq \varepsilon$. By (i), the run from c_1 to d reads from its pushdowns, at least once, a letter b with $b\,\mathcal{L} \cap X_0 \neq \emptyset$; we consider the first such transition t. If $t \in \Delta_\varepsilon$, the choice of t allows to prove that it can be executed at the very beginning (i.e., in the configuration c_1). Using that $\mathfrak{P}$ is saturated, the first two transitions can be combined into one of the form $(\overline{p}, a, u', \overline{q}')$ with $|u'| < |u|$ such that the induction hypothesis is applicable. If $t \notin \Delta_\varepsilon$, property (ii) implies that it is the very last transition (that leads to the configuration d) and that $a\,\mathcal{L} = b\,\mathcal{L}$. Using (ii), it follows that the very first transition can be postponed to the last-but-one position implying $c_0 \Vdash_{(X_1,\ldots,X_n)} \circ \vdash_\varepsilon^* \circ \Vdash_{X_{n+1}}^2 d$. □

It follows from the above lemma that $\vdash^*$ is the union of all relations $\Vdash_{\overline{X}}$ where the sets in the sequence $\overline{X}$ are mutually distinct implying that the length of $\overline{X}$ is bounded. Since the subsystems $\mathfrak{P}_\varepsilon$ and $\mathfrak{P}_X$ are homogenous, the arguments from Section 5.1 ensure that Theorem 5.1 holds for saturated systems.

5.3 Saturating a System

It remains to transform an arbitrary system into an equivalent saturated one. For a classical pushdown system (with just one pushdown), the idea is very simple: If there are transitions $(\overline{p}, a, bw, \overline{q})$ and $(\overline{q}, b, \varepsilon, \overline{r})$, then adding the transition $(\overline{p}, a, w, \overline{r})$ does not change the behavior and transforms the system closer to a saturated one. In the multi-pushdown setting, the technicalities are a bit more involved: Suppose we have the transitions $(\overline{p}, a, cbw, \overline{q})$ and $(\overline{q}, b, \varepsilon, \overline{r})$ with $c\,\mathcal{L} \cap b\,\mathcal{L} = \emptyset$, i.e., $b \parallel c$. Then $\overline{\pi}(cbw) = \overline{\pi}(bcw)$, i.e., after doing the first transition (that writes the trace $\overline{\pi}(cbw) = \overline{\pi}(bcw)$), the second transition (eliminating

b) can be executed immediately. Therefore, also in this situation, we add the transition $(\overline{p}, a, cw, \overline{r})$ to get closer to a saturated system.

Now, we construct cPDS $\mathfrak{P}^{(k)} = (\mathbf{Q}, \Delta^{(k)})$ for any $k \in \mathbb{N}$ as follows:

– we set $\Delta^{(0)} := \{(\overline{p}, a, \mathrm{lnf}(w), \overline{q}) \mid (\overline{p}, a, w, \overline{q}) \in \Delta\}$.[6]
– To obtain $\Delta^{(k+1)}$, we add to the set $\Delta^{(k)}$ all transitions $(\overline{p}, a, \mathrm{lnf}(uv), \overline{r})$ for which there are a letter $b \in A$ and a global state $\overline{q} \in \mathbf{Q}$ such that $(\overline{p}, a, ubv, \overline{q}), (\overline{q}, b, \varepsilon, \overline{r}) \in \Delta^{(k)}$ and $u \parallel b$.

Let $\Delta^{(\infty)} = \bigcup_{k\geq 0} \Delta^{(k)}$ be the "limit" of the increasing sequence of sets $\Delta^{(k)}$. Note that the length of words written by transitions in $\Delta^{(\infty)}$ is bounded by the length of words written by transitions in $\Delta^{(0)}$; hence $\Delta^{(\infty)}$ is finite.

Example 5.5. Recall the cPDS $\mathfrak{P}$ from Example 3.4. In Figure 6 we depict our construction of the multi-pushdown systems $\mathfrak{P}^{(k)}$ for $k \in \{0, 1, 2\}$. It can be verified that $\mathfrak{P}^{(2)} = \mathfrak{P}^{(3)}$.

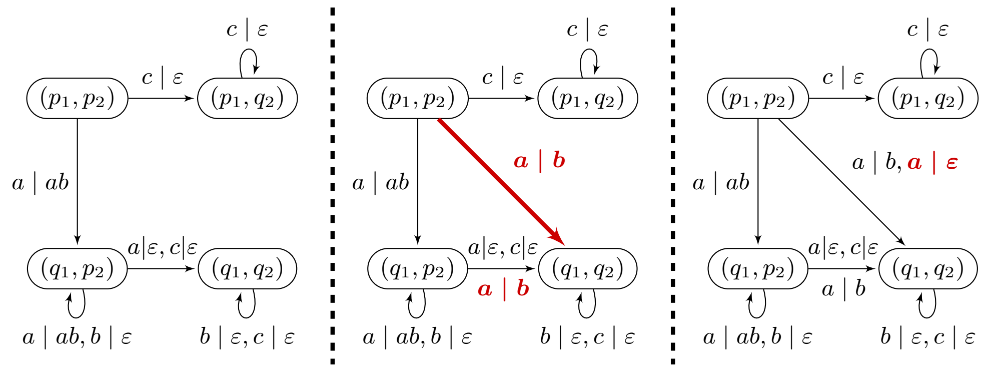
$c \mid \varepsilon$
(p_1, p_2)
(p_1, q_2)
(q_1, p_2)
(q_1, q_2)
$a \mid ab$
$a|\varepsilon, c|\varepsilon$
$a \mid ab, b \mid \varepsilon$
$b \mid \varepsilon, c \mid \varepsilon$
$\boldsymbol{a \mid b}$
$a \mid b, \boldsymbol{a \mid \varepsilon}$
$a \mid b$

Fig. 6. The multi-pushdown system $\mathfrak{P} = \mathfrak{P}^{(0)}$, $\mathfrak{P}^{(1)}$, and $\mathfrak{P}^{(2)} = \mathfrak{P}^{(\infty)}$ (from left to right). New transitions are marked in bold and red.

One can then show that the pair $\mathfrak{P}^{(\infty)} = (\mathbf{Q}, \Delta^{(\infty)})$ is a cPDS and that its reachability relation coincides with that of the original system $\mathfrak{P}$.

Proposition 5.6. *Let $\mathfrak{P}$ be a cPDS. Then, in time polynomial in $\|\mathfrak{P}\|^{|P|}$, one can construct an equivalent saturated system $\mathfrak{P}^{(\infty)}$, i.e., a saturated cPDS with $\vdash^*_{\mathfrak{P}} = \vdash^*_{\mathfrak{P}^{(\infty)}}$.*

[6] $\mathrm{lnf}(w)$ denotes the lexicographic normal form of the trace $\overline{\pi}(w)$. The use of $\mathrm{lnf}(w)$ instead of w allows to easily prove a polynomial upper bound for the number of transitions.

6 Summary, Consequences, and Open Questions

We proved that the backwards reachability relation of communicating multi-pushdown systems efficiently preserves the recognizability of a set of configurations and that the forwards reachability relation efficiently preserves rationality. Conversely, one can demonstrate that the backwards reachability relation does not preserve rationality (i.e., there is a cPDS and a rational set C of configurations such that $\text{pre}^*(C)$ is not rational anymore). Similarly, one can demonstrate that the forwards reachability relation does not preserve recognizability.

It is decidable whether a given rational set of traces is contained in a given recognizable set of traces. Hence our positive results allow to decide, for C_1 rational and C_2 recognizable, the following questions.

- $\text{post}^*(C_1) \subseteq C_2$, or, since the class of recognizable trace languages is closed under complementation, $\text{post}^*(C_1) \cap C_2 = \emptyset$. This amounts to a safety property.
  Since singleton sets are both recognizable and rational, this also implies that the reachability relation is decidable.
- $\text{post}^*(C_1) \cap C_2 \neq \emptyset$. Since the set C_2 of configurations with a given global state is recognizable, this implies that the control state reachability problem is decidable for C_1 rational.
- $C_1 \subseteq \text{pre}^*(C_2)$.
- $\text{post}^*(C_1) \subseteq \text{pre}^*(C_2)$ which amounts to a liveness property: From every configuration reachable from C_1, we can reach a configuration from C_2. This property can also by expressed by the EF-formula $C_1 \wedge \neg\text{EF}\,(\neg\text{EF}\,C_2)$. More generally, EF-model checking is decidable for cPDS, although our results allow to bound the running time only non-elementary.

The next and obvious open question regarding the verification of cPDS, one would have to consider the recurrent reachability, i.e., the question whether, starting from some configuration, there is an infinite run that visits some global state infinitely often. This could then form the basis for algorithms deciding properties that are given by formulas from linear time temporal logics.

Since we can see cPDS as a natural extension of pushdown systems from word semantics to trace semantics, another open problem is to find some generalized context-free grammars accepting the class of languages of cPDS. Additionally, one could compare this new model with other known models for multi-pushdown systems.

References

1. Aiswarya, C., Gastin, P., Narayan Kumar, K.: Controllers for the verification of communicating multi-pushdown systems. In: CONCUR'14. pp. 297–311. Lecture Notes in Computer Science vol. 8704, Springer (2014)
2. Aiswarya, C., Gastin, P., Narayan Kumar, K.: Verifying communicating multi-pushdown systems via split-width. In: ATVA'14. pp. 1–17. Lecture Notes in Computer Science vol. 8837, Springer (2014)

3. Atig, M.F., Bollig, B., Habermehl, P.: Emptiness of ordered multi-pushdown automata is 2etime-complete. Int. J. Found. Comput. Sci. **28**(8), 945–976 (2017)
4. Babic, D., Rakamaric, Z.: Asynchronously communicating visibly pushdown systems. In: FMOODS/FORTE'13. pp. 225–241. Lecture Notes in Computer Science vol. 7892, Springer (2013)
5. Bollig, B., Kuske, D., Mennicke, R.: The complexity of model checking multi-stack systems. Theory of Computing Systems **60**(4), 695–736 (2017)
6. Bouajjani, A., Esparza, J., Maler, O.: Reachability analysis of pushdown automata: Application to model-checking. In: Mazurkiewicz, A., Winkowski, J. (eds.) 8th International Conference on Concurrency Theory. Lecture Notes in Computer Science, vol. 1243, pp. 135–150. Springer (1997)
7. Bouajjani, A., Esparza, J., Touili, T.: A generic approach to the static analysis of concurrent programs with procedures. Int. J. Found. Comput. Sci. **14**(4), 551 (2003)
8. Bouajjani, A., Müller-Olm, M., Touili, T.: Regular Symbolic Analysis of Dynamic Networks of Pushdown Systems. In: Abadi, M., de Alfaro, L. (eds.) CONCUR 2005 – Concurrency Theory. pp. 473–487. Lecture Notes in Computer Science vo. 3653, Springer (2005)
9. Cartier, P., Foata, D.: Problemes combinatoires de commutation et rearrangements. Lecture Notes in Mathematics vol. 85, Springer, Berlin - Heidelberg - New York (1969)
10. Diekert, V., Rozenberg, G.: The Book of Traces. World scientific (1995)
11. Finkel, A., Willems, B., Wolper, P.: A Direct Symbolic Approach to Model Checking Pushdown Systems. Electronic Notes in Theoretical Computer Science **9**, 27–37 (1997)
12. Heußner, A., Leroux, J., Muscholl, A., Sutre, G.: Reachability analysis of communicating pushdown systems. Log. Methods Comput. Sci. **8**(3) (2012)
13. Hutagalung, M., Hundeshagen, N., Kuske, D., Lange, M., Lozes, É.: Multi-buffer simulations: decidability and complexity. Information and Computation **262**(2), 280–310 (2018)
14. Kleene, S.: Representation of events in nerve nets and finite automata. In: Shannon, C., McCarthy, J. (eds.) Automata Studies, pp. 3–40. Annals of Mathematics Studies vol. 34, Princeton University Press (1956)
15. La Torre, S., Madhusudan, P., Parlato, G.: A robust class of context-sensitive languages. In: LICS'07. pp. 161–170. IEEE Computer Society (2007)
16. La Torre, S., Napoli, M., Parlato, G.: Reachability of scope-bounded multistack pushdown systems. Inf. Comput. **275**, 104588 (2020)
17. Mazurkiewicz, A.: Concurrent program schemes and their interpretations. DAIMI Report Series **6**(78) (1977)
18. Qadeer, S., Rehof, J.: Context-bounded model checking of concurrent software. In: TACAS'05. pp. 93–107. Lecture Notes in Computer Science vol. 3440, Springer (2005)
19. Zielonka, W.: Notes on finite asynchronous automata. RAIRO - Theoretical Informatics and Applications **21**(2), 99–135 (1987)