# Reachability problem for weak multi-pushdown automata

Wojciech Czerwiński, Piotr Hofman, and Sławomir Lasota*

Institute of Informatics, University of Warsaw
wczerwin,ph209519,sl@mimuw.edu.pl

**Abstract.** This paper is about reachability analysis in a restricted subclass of multi-pushdown automata: we assume that the control states of an automaton are partially ordered, and all transitions of an automaton go downwards with respect to the order. We prove decidability of the reachability problem, and computability of the backward reachability set. As the main contribution, we identify relevant subclasses where the reachability problem becomes NP-complete. This matches the complexity of the same problem for communication-free vector addition systems (known also as commutative context-free graphs), a special case of stateless multi-pushdown automata.

## 1 Introduction

This paper is about reachability analysis of *multi-pushdown systems*, i.e., systems with a global control state and multiple stacks. The motivation for our work is twofold. On one side, a practical motivation coming from context-bounded analysis of recursive concurrent programs [23, 20, 3]. On the other side, a theoretical motivation coming from partially-commutative context-free grammars, developed recently in [11–13].

*Context bounded analysis.* Multi-pushdown systems may be used as an abstract model of concurrent programs with recursive procedures. As multi-pushdown systems are a Turing-complete model of computation, they are only applicable for verification under further tractable restrictions. One remarkably successful restriction is imposing a bound on the number of context switches; between consecutive context switches, the system may only perform operations on one stack (local operations). In [23], the context-bounded reachability has been shown decidable, by reduction to reachability of ordinary pushdown systems [5]. This line of research, with applications in formal verification, has been continued successfully, e.g., in [6, 20, 3].

*Weak control states.* As our starting point we observe that if the number of context switches is bounded, one may safely assume that the control state space is *weak*, in the sense that there is a partial order on control states such that transitions go only downwards with respect to the order. Indeed, the local state space of every thread may be eliminated using a stack, and the global control state essentially enumerates context

* The first author acknowledges a partial support by the Polish MNiSW grant N N206 568640. The other authors acknowledge a partial support by the Polish MNiSW grant N N206 567840.

switches. Roughly speaking, the model investigated in this paper extends the above one with respect to operations allowed between two context switches, namely, we do not restrict these operations to one stack only. Thus, if $k$ is the number of stacks, we assume that transitions of a system are of the following form:

$$q,\ \mathrm{X} \xrightarrow{a} q',\ \alpha_1, \ldots, \alpha_k, \tag{1}$$

to mean that in state $q$, symbol X is popped from one of the stacks, and sequences of symbols $\alpha_1, \ldots, \alpha_k$, respectively, are pushed on stacks. Wlog. one may assume that the symbols of different stacks are different.

*Partially-commutative context-free grammars.* A special case of the model investigated in this paper is *stateless* multi-pushdown systems. This is still a quite expressible model that subsumes, among the others, context-free graphs (so called Basic Process Algebra [8]) and communication-free Petri nets (so called Basic Parallel Processes [8]). In the stateless case, transitions (1) may be understood as productions of a grammar, with the nonterminal symbols on the right-hand side (stack symbols) subject to a commutativity law. More precisely, for any two symbols X and Y from different stacks, we impose the commutativity law

$$\mathrm{XY} \ = \ \mathrm{YX}.$$

One easily observes that this is a special case of *independence relation* over nonterminal symbols, as defined in trace theory [14][1]. In multi-pushdown systems, the *dependency relation* (complement of independence relation) is always transitive. A general theory of context-free grammars modulo dependency relation that is not necessarily transitive, has been studied recently in [13]; complexity of bisimulation equivalence checking has been investigated in [11, 12]. The present paper complements these results by focusing on reachability analysis.

*Contributions.* This paper contains two main results. First, we prove decidability of reachability for weak multi-pushdown automata. Our argument is based on a suitable well order on the set of configurations, that strongly depends on the assumption that the control states are weak.

Second, we identify additional restrictions under which the problem is NP-complete; one such restriction is stateless multi-pushdown systems. Our result subsumes (and gives a simpler algorithm for) the case of communication-free Petri nets; reachability thereof is NP-complete as shown in [15]. The last result is similar to NP-completeness of the word problem for partially-commutative context-free grammars [16], where one asks if the given input word is accepted. The reachability question is more difficult to answer, as an input word is not given in advance. In fact the main technical difficulty is to show existence of a polynomial witness for reachability.

As further results, we investigate forward and backward reachability sets, and prove that the backward reachability set of a regular set of configurations is regular and computable, while the forward reachability set needs not be regular in general. Finally,

[1] Note however that the independence is imposed on nonterminal symbols, and not on input letters, as usually in trace theory.

we identify the decidability border for reachability of weak multi-pushdown systems. Roughly speaking, the problem becomes undecidable when one asks about reachability of a given regular set of configurations, instead of a single configuration.

The standard techniques useful for analysis of pushdown systems, such as pumping or the automaton-based approach of [5], do not extend to the multi-pushdown setting. This is why the proofs of our results are based on new insights. The NP-membership proofs are, roughly speaking, based on polynomial witnesses obtained by careful elimination of 'irrelevant' transitions. On the other hand, the decidability results are based on a suitable well order on configurations.

*Related research.* Multi-pushdown systems are a fundamental model of recursive multi-threaded programs. This is why different instantiations of the multi-pushdown paradigm have been appearing in the literature recently, most often in the context of formal verification. We only mention here a few relevant positions we are aware of, without claiming completeness. All the papers cited below bring some restricted decidability results for reachability or model checking.

Most often, a model has global control states, subject to some restriction. For instance, the author of [1] assumes that the stacks are ordered, and pop operation can only be performed on the first nonempty stack. Another example is the model introduced in [7] and then further investigated e.g. in [6, 2, 4], that allows for unbounded creation of new stacks; on the other hand, operations on each stack are local, thus no communication between threads is allowed.

Another possible approach is to replace global state space with some communication mechanism between threads. Some successful results on analysis of multi-threaded programs communicating via locks, in a restricted way, has been reported in [18, 17, 9].

In [21] the algorithm for reachability over PA [8] graphs has been provided. The PA class is a generalization of both BPA and BPP that allows, similarly like multi-pushdown systems, both for sequential and interleaved behavior. Finally, in [19] the reachability problem has been shown decidable for Process Rewrite Systems [22] extended with weak control states.

*Outline.* In the following Section 2 we define the model we work with. Then in Section 3 we state all our results. In the remaining sections we provide proofs of some of the results. The other proofs are omitted due to space limitation.

## 2 Multi-pushdown automata

A multi-pushdown automaton (MPDA) is like a single-pushdown one. In a single step one symbol is popped from one of stacks[2], and a number of symbols are pushed on the stacks. Assume there is $k$ stacks. A transition of an automaton is thus of the form:

$$q,\ \mathrm{X} \xrightarrow{a} q',\ \alpha_1, \ldots, \alpha_k, \tag{2}$$

[2] If we allowed for popping from more than one stack at a time, the model would clearly become Turing-complete, even with 1 state only.

to mean that when an automaton reads $a$ in state $q$, it pops X from one of the stacks, pushes the sequence of symbols $\alpha_i$ on the $i$th stack, for $i = 1 \dots k$, and goes to state $q'$. We allow for silent transitions with $a = \varepsilon$. Observe that wlog. one may assume that stack alphabets are disjoint.

Formally, the ingredients of a MPDA are: a finite set of states $\mathsf{Q}$, the number of stacks $k$, pairwise-disjoint finite stack alphabets $\mathsf{S_1} \dots \mathsf{S_k}$, an input alphabet $\mathsf{A}$, and a finite set of transition rules:

$$\longrightarrow \subseteq \mathsf{Q} \times (\bigcup_{i \leq k} \mathsf{S_i}) \times (\mathsf{A} \cup \{\varepsilon\}) \times \mathsf{Q} \times \mathsf{S_1}^* \times \ldots \times \mathsf{S_k}^* \tag{3}$$

written as in (2). A configuration of a MPDA is a tuple $\langle q, \beta_1, \dots, \beta_k \rangle \in \mathsf{Q} \times \mathsf{S_1}^* \times \ldots \times \mathsf{S_k}^*$. The transition rules (2) induce the transition relation over all configurations in a standard way:

$$\frac{q,\ \mathrm{X} \xrightarrow{a} q',\ \alpha_1, \dots, \alpha_k \qquad \mathrm{X} \in \mathsf{S_i} \qquad \beta_i = \mathrm{X}\beta}{\langle q, \beta_1, \dots \beta_i \dots, \beta_k \rangle \xrightarrow{a} \langle q', \alpha_1\beta_1, \dots \alpha_i\beta \dots, \alpha_k\beta_k \rangle}$$

thus defining the configuration graph of a MPDA. For a configuration $\langle q, \alpha_1, \dots, \alpha_k \rangle$, its *size* is defined as the sum of lengths of the words $\alpha_i$, $i \leq k$. The same applies to a right-hand side of any transition rule $q\,\mathrm{X} \xrightarrow{a} q'\,\alpha_1 \dots \alpha_k$.

An MPDA is *stateless* if there is just one state (or equivalently no states). Transition rules of an automaton are then of the form:

$$\mathrm{X} \xrightarrow{a} \alpha_1, \dots, \alpha_k \tag{4}$$

and configurations are of the form $\langle \beta_1, \dots, \beta_k \rangle$.

A less severe restriction on control states is the following one. We say that an automaton is *weak* if there is a partial order $\leq$ on its states such that every transition (2) satisfies $q' \leq q$. Clearly, every stateless automaton is weak.

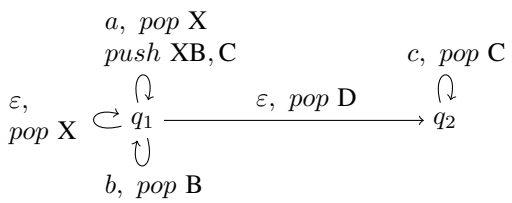
*Remark 1.* Note that stateless one-stack automata are essentially context-free grammars in Greibach normal form. Thus the configuration graphs are precisely context-free graphs, called also BPA graphs [22, 8]. Another special case is many stacks with singleton alphabets. The stacks are thus essentially counters without zero tests. In this subclass, stateless automata corresponds to communication-free Petri nets [15], called also BPP [10], or commutative context-free graphs [12]. The BPA and BPP classes are members of the Process Rewrite Systems hierarchy of [22] that contains, among the others, pushdown systems and unrestricted Petri nets.

**Example 1.** Assuming a distinguished initial state and acceptance by all stacks empty, weak MPDAs can recognize non-context-free languages. For instance, the language

$$\{a^n b^n c^n : n \geq 0\} \tag{5}$$

is recognized by an automaton described below. The automaton has two states $q_1, q_2$ and two stacks. The alphabets of the stacks are $\{\mathrm{X}, \mathrm{B}, \mathrm{D}\}$ and $\{\mathrm{C}\}$, respectively. The

starting configuration is $(q_1, \mathrm{XD}, \varepsilon)$. Besides the transition rules, we also present the automaton in a diagram, using *push* and *pop* operations with natural meaning.




$$
\begin{aligned}
q_1,\ \mathrm{X} &\xrightarrow{a} q_1,\ \mathrm{XB},\ \mathrm{C}\\
q_1,\ \mathrm{X} &\xrightarrow{\varepsilon} q_1,\ \varepsilon,\ \varepsilon\\
q_1,\ \mathrm{B} &\xrightarrow{b} q_1,\ \varepsilon,\ \varepsilon\\
q_1,\ \mathrm{D} &\xrightarrow{\varepsilon} q_2,\ \varepsilon,\ \varepsilon\\
q_2,\ \mathrm{C} &\xrightarrow{c} q_2,\ \varepsilon,\ \varepsilon
\end{aligned}
$$

The automaton is weak and uses $\varepsilon$-transitions, which may be however easily eliminated. Acceptance by empty stacks may be easily simulated using acceptance by states. The language (5) is not recognized by a stateless automaton, as shown in [13].

**Example 2.** Non-context-free languages are recognized even by stateless MPDAs with singleton stack alphabets. The class of languages recognized by this subclass is called *commutative context-free* languages [16], see also [13]. One example is the commutative closure of the language of the previous example: the set of all words with the same number of occurrences of $a$, $b$ and $c$.

In the sequel we do not care about initial states nor about acceptance condition, as we will focus on the configuration graph of an automaton. Furthermore, as we only consider reachability problem, the labeling of transitions with input alphabet letters will be irrelevant, thus we write $\longrightarrow$ instead of $\xrightarrow{a}$ from now on.

Using a standard terminology, we say that a MPDA is *normed* if for any state $q$ and any configuration $\langle q, \alpha_1, \ldots, \alpha_k \rangle$, there is a path to the empty configuration

$$\langle q, \alpha_1, \ldots, \alpha_k \rangle \longrightarrow \ldots \longrightarrow \langle p, \varepsilon, \ldots, \varepsilon \rangle$$

for whatever state $p$. In general, whenever a MPDA is not assumed to be normed we call it *unnormed* for clarity. Note that in all examples above the automata were normed. In fact normedness is not a restriction as far as languages are considered. In the sequel we will however analyze the configuration graphs, and then normedness will play a role.

Further, we say that a MPDA is *strongly normed* if for any state $q$ and any configuration $\langle q, \alpha_1, \ldots, \alpha_k \rangle$, there is a path to the empty configuration

$$\langle q, \alpha_1, \ldots, \alpha_k \rangle \longrightarrow \ldots \longrightarrow \langle q, \varepsilon, \ldots, \varepsilon \rangle$$

containing only transitions that do not change state. Intuitively, whatever is the state $q$ we start in, any top-most symbol X in any stack may „disappear". For stateless automata, strong normedness is the same as normedness.

## 3 Reachability

**Regular sets.** We will consider various reachability problems in the configuration graph of a given MPDA. Therefore, we need a finite way of describing infinite sets

of configurations. A standard approach is to consider *regular* sets. Below we adapt this approach to the multi-stack scenario we deal with.

Consider the configurations of a stateless MPDA, $\mathsf{S} = \mathsf{S_1}^* \times \ldots \times \mathsf{S_k}^*$. There is a natural monoid structure in $\mathsf{S}$, with pointwise identity $\langle \varepsilon, \ldots, \varepsilon \rangle$ and multiplication

$$\langle \alpha_1, \ldots, \alpha_k \rangle \cdot \langle \beta_1, \ldots, \beta_k \rangle \;=\; \langle \alpha_1\, \beta_1, \ldots, \alpha_k\, \beta_k \rangle.$$

Call a subset $L \subseteq \mathsf{S}$ *regular* if there is a finite monoid $M$ and a monoid morphism $\gamma : \mathsf{S} \to M$ that *recognizes* $L$, which means that $L = \gamma^{-1}(N)$ for some subset $N \subseteq M$. Without loss of generality one may assume that the monoid $M$ is a product of finite monoids $M = M_1 \times \ldots \times M_k$, and that

$$\gamma = \gamma_1 \times \ldots \times \gamma_k \quad \text{where} \quad \gamma_i : \mathsf{S_i}^* \to M_i \;\text{ for } i = 1 \ldots k.$$

Thus we may use an equivalent but more compact representation of regular sets, based on automata: a regular set $L$ is given by a tuple of (nondeterministic) finite automata $\mathcal{B}_1 \ldots \mathcal{B}_k$ over alphabets $\mathsf{S_1} \ldots \mathsf{S_k}$, respectively, together with a set

$$F \subseteq Q_1 \times \ldots \times Q_k$$

of accepting tuples of states, where $Q_i$ denotes the state space of automaton $\mathcal{B}_i$.

Unless stated otherwise, in the sequel we always use such representations of regular sets of configurations. If there are more than one state, we assume a representation for every state. In particular, when saying "polynomial wrt. $L$", for a regular language $L$, we mean polynomial wrt. the sum of sizes of automata representing $L$.

*Remark 2.* Clearly, the cardinality of the set $F$ of accepting tuples may be exponential wrt. the cardinalities of state spaces of automata $\mathcal{B}_i$. However, complexities we derive in the sequel will never depend on cardinality of $F$.

**Example 3.** Assume that there are two stacks. An example of properties we can define is: „odd number of elements on the first stack and symbol A on the top of the second stack, or an even number of the elements on the first stack and the odd number of elements on the second stack". On the other hand, "all stacks have equal size" is not a regular property according to our definition.

*Remark 3.* We have deliberately chosen a notion of regularity of languages of *tuples* of words. Another possible approach could be to consider regular languages of words, over the product alphabet $(\mathsf{S_1} \cup \bot) \times \ldots \times (\mathsf{S_k} \cup \bot)$, where the additional symbol $\bot$ is necessary for padding. This would yield a larger class, for instance the last language from Example 3 would be regular. The price to pay would be however undecidability of the reachability problems. The undecidability will be discussed below.

**Reachability.** In this paper we consider the following reachability problem:

INPUT: a MPDA $\mathcal{A}$ and two regular sets of configurations $L, K \subseteq \mathsf{S}$.
QUESTION: is there a path in the configuration graph from $L$ to $K$?

We will write $L \rightsquigarrow_{\mathcal{A}} K$ if a path from $L$ to $K$ exists in the automaton $\mathcal{A}$. The sets $L$ and $K$ we call *source* and *target* sets, respectively. We will distinguish special cases, when either $L$ or $K$ or both the sets are singletons, thus obtaining four different variants of reachability altogether. For brevity we will use symbol '*1*' for a singleton, and symbol 'REG' for a regular set, and speak of *1*$\rightsquigarrow$REG *reachability* (when $L$ is a singleton), REG$\rightsquigarrow$REG *reachability* (the unrestricted case), and likewise for REG$\rightsquigarrow$*1* and *1*$\rightsquigarrow$*1*.

Before stating the results, we note that all the problems we consider here are NP-hard:

**Lemma 1.** *The 1$\rightsquigarrow$1 reachability is NP-hard for strongly normed stateless MPDAs, even if all stack alphabets are singletons.*

The above fact follows immediately from NP-completeness of the reachability problem for communication-free Petri nets, see [15] for details.

**Results.** In presence of states, the 1$\rightsquigarrow$1 reachability problem is obviously undecidable, because the model is Turing powerful. Undecidability holds even for normed MPDAs. We will thus consider only stateless or weak MPDAs from now on.

We start by observing that out of four combinations of the reachability problem, it is sufficient to consider only two, namely the REG$\rightsquigarrow$1 and REG$\rightsquigarrow$REG cases. Indeed, as far as complexity is concerned, we observe the following collapse:

$$1\rightsquigarrow 1 \;=\; \text{REG}\rightsquigarrow 1 \qquad\qquad 1\rightsquigarrow \text{REG} \;=\; \text{REG}\rightsquigarrow\text{REG} \tag{6}$$

independently of a restriction on automata. The first equality follows from our first result:

**Lemma 2.** *Suppose $\mathcal{A}$ is a weak MPDA. Let $L$ be a regular set of configurations of $\mathcal{A}$ and let $t$ be a configuration of $\mathcal{A}$. Then*

$$L \rightsquigarrow_{\mathcal{A}} t \implies s \rightsquigarrow_{\mathcal{A}} t \text{ for some } s \in L \text{ of size polynomial wrt. } \mathcal{A}, L \text{ and } t.$$

Indeed, the reduction from REG$\rightsquigarrow$1 to 1$\rightsquigarrow$1 is by nondeterministic guessing a source configuration of polynomial size.

The second equality (6) will follow from our results listed below.

Before stating the remaining results, we summarize all of them in the following table. We distinguish cases, corresponding to strongly normed/normed/unnormed case and stateless/weak case. Each entry of the table contains the complexity of REG$\rightsquigarrow$REG reachability problem. Additionally, the complexity of REG$\rightsquigarrow$1 reachability problem is given in cases it is different from the complexity of REG$\rightsquigarrow$REG reachability.

For clarity, we do not distinguish stateless strongly normed case from stateless normed one, as these two cases obviously coincide.

| [ REG$\rightsquigarrow$1]<br>REG$\rightsquigarrow$REG | strongly normed | normed | unnormed |
|---|---|---|---|
| stateless | NP-compl. (Thm. 2) | | [ NP-compl. (Thm. 3) ]<br>undecidable (Thm. 1) |
| weak | NP-compl. (Thm. 2) | [ decidable ]<br>undecidable (Thm. 1) | [ decidable (Thm. 4) ]<br>undecidable |

Now we discuss the results in detail. We first observe an apparent decidability frontier witnessed by stateless unnormed MPDAs and weak normed MPDAs:

**Theorem 1.** *The $1 \rightsquigarrow \mathrm{REG}$ reachability is undecidable for stateless unnormed MPDAs and for weak normed MPDAs.*

The proof is by reduction of the nonemptiness of intersection of context-free languages and uses three stacks. The case of two stacks remains open.

Thus lack of strong normedness combined with a regular target set yields undecidability in case of stateless automata. Surprisingly, restricting additionally:

- either the automaton to be strongly normed,
- or the target set to a singleton,

makes a dramatical difference for complexity of the problem, as summarized in Theorems 2, 3 and 4 below. In the first theorem we only assume strong normedness:

**Theorem 2.** *The $\mathrm{REG} \rightsquigarrow \mathrm{REG}$ reachability is NP-complete for strongly normed weak MPDAs.*

Theorem 2 is the main result of this paper. It is proved by showing that reachability is always witnessed by a polynomial witness, obtained by careful elimination of 'irrelevant' transitions.

In the following two theorems we do not assume strong normedness, thus according to Theorem 1 we have to restrict target set to singleton. Under such a restriction, we are able to prove NP-completeness only in the class of stateless MPDA, while for all weak MPDA we merely state decidability:

**Theorem 3.** *The $\mathrm{REG} \rightsquigarrow 1$ reachability is NP-complete for stateless unnormed MPDAs.*

**Theorem 4.** *The $\mathrm{REG} \rightsquigarrow 1$ reachability is decidable for weak unnormed MPDAs.*

Theorem 3 is shown similarly to Theorem 2, while the proof of Theorem 4 is based on a well order, the point-wise extension of a variant of Higman ordering.

**Open questions.** Except for two entries in the summarizing table above, we know the exact complexity of the reachability problem. The important open question that remains is the actual complexity of $1 \rightsquigarrow 1$ reachability for (normed and unnormed) weak MPDAs. Another interesting question is whether undecidability carries over to automata with two stacks only.

**Reachability set.** Now we consider the problem of computing the whole reachability set. For a given automaton $\mathcal{A}$, and a set $L$ of configurations, we consider forward and backward reachability sets of $L$, defined as:

$$\{s : L \rightsquigarrow_{\mathcal{A}} s\} \qquad \text{and} \qquad \{s : s \rightsquigarrow_{\mathcal{A}} L\},$$

respectively. It turns out that the backward reachability set may be computed under the strong normedness assumption.

**Theorem 5.** *For weak strongly normed MPDAs, the backward reachability set of a regular set is an effectively computable regular set.*

Roughly speaking, we show that the backward reachability set is upward closed with respect to the point-wise extension of a suitable variant of Higman ordering.

On the other hand, the forward reachability set needs not be regular, even in the case of strongly normed stateless automata, as shown in the following example.

**Example 4.** Consider a stateless automaton with two stacks, over alphabets $\{\mathrm{A}, \mathrm{X}\}$ and $\{\mathrm{B}\}$, and the following transition rules:

$$\mathrm{X} \longrightarrow \mathrm{XA},\ \mathrm{B} \qquad \mathrm{X} \to \varepsilon,\ \varepsilon \qquad \mathrm{A} \to \varepsilon,\ \varepsilon \qquad \mathrm{B} \to \varepsilon,\ \varepsilon.$$

The set of configurations reachable from the configuration $(\mathrm{X}, \varepsilon)$ is not regular:

$$\{(\mathrm{A}^i, \mathrm{B}^j) : i, j \in \mathbb{N}\} \ \cup\ \{(\mathrm{XA}^k, \mathrm{B}^l) : k \geq l\}.$$

**Relaxed regularity.** The relaxed definition of regularity, as discusses in Remark 3, makes the reachability problem intractable in all cases. The following theorem is shown by reduction from the Post Correspondence Problem:

**Theorem 6.** *The $1{\rightsquigarrow}\mathrm{REG}$ reachability is undecidable for stateless strongly normed MPDAs, under the relaxed notion of regularity.*

Furthermore, the backward reachability set of a relaxed regular set is not necessarily regular, even in stateless strongly normed MPDAs. The illustrating example is omitted due to space limitation.

## 4 Proof of Lemma 2

Consider a MPDA $\mathcal{A}$ and a regular set $L$ of configurations of $\mathcal{A}$. Let $s \in L$ be source configuration and let $t$ be an arbitrary target configuration. Suppose $s \rightsquigarrow_{\mathcal{A}} t$. We will show that the size of $s$ may be reduced, while preserving membership in $L$. The crucial but simple idea of the proof will rely on an analysis of *relevance* of symbol occurrences, to be defined below.

*Symbol occurrences.* Suppose that there is a path $\pi$ from $s$ to $t$, consisting of consecutive transitions $s \longrightarrow s_1 \longrightarrow s_2 \ldots \longrightarrow s_n = t$. We will consider all individual occurrences of symbols that appear in the configurations. For instance, in the following exemplary sequence of two-stack configurations

$$\langle q, \mathrm{AA}, \mathrm{C}\rangle \longrightarrow \langle q, \mathrm{BBA}, \mathrm{DC}\rangle \longrightarrow \langle q, \mathrm{ABBA}, \mathrm{DC}\rangle \tag{7}$$

there are altogether 14 *symbol occurrences*: 3 in the first configuration, 5 in the second one and 6 in the third one.

Recall that every transition $s_i \longrightarrow s_{i+1}$ is induced by some transition rule $\mathrm{X} \longrightarrow \alpha$ of the automaton. Then there is a distinguished occurrence of symbol X in $s_i$ that is

involved in the transition. In the sequel we use the term *symbol occurrence involved in a transition*.

Precisely one occurrence of symbol in $s_i$ is involved in the transition $s_i \longrightarrow s_{i+1}$; for every other occurrence of a symbol in $s_i$ there is a *corresponding* occurrence of the same symbol in $s_{i+1}$. (Note that we always make a difference between corresponding symbol occurrences from different configurations.) All remaining occurrences of symbols in $s_{i+1}$ are created by the transition; we call these occurrences *fresh*.

We define the *descendant* relation as follows. All fresh symbol occurrences in $s_{i+1}$ are descendants of the symbol occurrence in $s_i$ involved in the transition $s_i \longrightarrow s_{i+1}$. Moreover, a symbol occurrence in $s_{i+1}$ corresponding to a symbol occurrence in $s_i$ is its descendant too. We will use term *descendant* for the reflexive-transitive closure of the relation defined above and the term *ancestor* for its inverse relation. In particular, every symbol occurrence in $t$ is descendant of a unique symbol occurrence in $s$. The descendant relation is a forest, i.e., a disjoint union of trees.

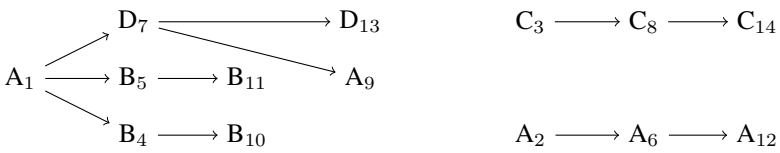
**Example 5.** As an example, consider again the sequence of transitions (7), with symbol occurrences identified by subscripts $1 \ldots 14$:

$$\langle q, \mathrm{A}_1\mathrm{A}_2, \mathrm{C}_3\rangle \longrightarrow \langle q, \mathrm{B}_4\mathrm{B}_5\mathrm{A}_6, \mathrm{D}_7\mathrm{C}_8\rangle \longrightarrow \langle q, \mathrm{A}_9\mathrm{B}_{10}\mathrm{B}_{11}\mathrm{A}_{12}, \mathrm{D}_{13}\mathrm{C}_{14}\rangle \quad (8)$$

Say the transitions are induced by the following two transition rules:

$$q,\ \mathrm{A} \longrightarrow q,\ \mathrm{BB},\ \mathrm{D} \qquad\qquad q,\ \mathrm{D} \longrightarrow q,\ \mathrm{A},\ \mathrm{D}$$

The descendant relation can be presented as the following forest:




The symbol occurrences involved in the two transitions (8) are $\mathrm{A}_1$ in the first configuration and $\mathrm{D}_7$ in the second one. The fresh symbol occurrences are $\mathrm{B}_4$, $\mathrm{B}_5$ and $\mathrm{D}_7$ in the second configuration, and $\mathrm{A}_9$ and $\mathrm{D}_{13}$ in the third one.

*Relevant symbol occurrences.* As the automaton $\mathcal{A}$ is weak, the number of transitions in $\pi$ that change state is bounded by the number of states of $\mathcal{A}$. All remaining transitions in $\pi$ do not change state.

Consider all the occurrences of all symbols in all configurations along the path $\pi$, including configurations $s$ and $t$ themselves. A symbol occurrence is called *relevant* if some of its descendants:

- belongs to the target configuration $t$; or
- is involved in some transition in $\pi$ that changes state.

Otherwise, a symbol occurrences is *irrelevant*. In particular, all symbol occurrences in $t$ are relevant. Referring back to our example, all symbol occurrences appearing in (8) are relevant.

Note that if $t$ is not the empty configuration then every configuration in $\pi$ contains at least one relevant symbol occurrence. On the other side, in every configuration, the number of relevant occurrences is always bounded by the sum of the size of $t$ and the number of states of $\mathcal{A}$.

*Small source configuration.* So prepared, we are ready to prove that there is a configuration $s' \in L$ of polynomial size with $s' \rightsquigarrow_{\mathcal{A}} t$. We will rely on the following claim:

**Lemma 3.** *For any configuration $s'$ obtained from $s$ by removing some irrelevant symbol occurrences, it holds $s' \rightsquigarrow_{\mathcal{A}} t$.*

The lemma follows from the following two observations: (1) all the transitions in $\pi$ involving symbol occurrences remaining in $s'$ and their descendants may be re-done; (2) the resulting configuration will be exactly $t$, as only irrelevant symbol occurrences have been removed from $s$.

Recall that the language $L$ is represented by a tuple $\mathcal{B}_1 \dots \mathcal{B}_k$ of deterministic finite automata, one automaton per stack. Consider the content of a fixed $i$th stack in $s$, say $w \in \mathsf{A}_i^*$. Let $n$ be the number of states of $\mathcal{B}_i$. The run of the automaton $\mathcal{B}_i$ over $w$ labels every position of $w$ by some state. We will use a standard pumping argument to argue that every block of consecutive irrelevant symbol occurrences in $s$ may be reduced in length to at most $n$. Indeed, upon every repetition of a state of $\mathcal{B}_i$, the word $w$ may be shortened by removing the induced infix, while preserving membership in $L$. By repeating the pumping argument for all blocks of consecutive irrelevant symbol occurrences in all stacks in $s$, one obtains a configuration $s'$, still belonging to $L$, of quadratic size. By Lemma 3 we know that $s' \rightsquigarrow t$, as required.

## 5 Proof of Theorem 2

NP-hardness follows from Lemma 1. The proof of membership in NP relies on the following two core lemmas:

**Lemma 4.** *The 1$\rightsquigarrow$1 reachability problem is in NP for strongly normed weak MPDAs.*

**Lemma 5.** *Let $\mathcal{A}$ be a strongly normed weak MPDA and let $L, K$ be regular sets of configurations. If $L \rightsquigarrow K$ then $s \rightsquigarrow t$ for some $s \in L$ and $t \in K$ of size polynomial wrt. the sizes of $\mathcal{A}$, $L$ and $K$.*

The two lemmas easily yield a decision procedure for REG$\rightsquigarrow$REG reachability: guess configurations $s \in L$ and $t \in K$ of size bounded by a polynomial deduced from the proof of Lemma 5, and then apply the procedure of Lemma 4 to check if $s \rightsquigarrow t$.

The rest of this section is devoted to the part of the proof of Lemma 4. The remaining part of the proof, together with the proof of Lemma 5, are omitted.

### 5.1 Proof of Lemma 4

Consider a MPDA $\mathcal{A}$ and two configurations $s$ and $t$. We will define a nondeterministic polynomial-time decision procedure for $s \rightsquigarrow_{\mathcal{A}} t$.

*Stateless assumption.* For simplicity, we assume that both $s$ and $t$ have the same control state. Thus we can treat transitions that lead from $s$ to $t$ as stateless transitions. At the very end of the proof, we will discuss how to generalize it to the general case of strongly normed weak MPDAs.

*Polynomial witness.* Our aim is to show that if there is a path from $s$ to $t$ then there is a path of polynomial length. So stated, the above claim may not be verbally true, even in the case of context-free graphs, as witnessed by the following simple example.

$$\mathrm{X}_1 \longrightarrow \mathrm{X}_2\mathrm{X}_2 \qquad \mathrm{X}_2 \longrightarrow \mathrm{X}_3\mathrm{X}_3 \qquad \ldots \qquad \mathrm{X}_{n-1} \longrightarrow \mathrm{X}_n\mathrm{X}_n \qquad \mathrm{X}_n \longrightarrow \varepsilon \tag{9}$$

The example scales with respect to $n$, and thus the shortest path from the configuration $\mathrm{X}_1$ to $\mathrm{X}_n$ is of exponential length. As a conclusion, one must use some subtle analysis in order to be able to reduce the length of a witness of existence of the path as required. Note that $\mathrm{X}_1$ is relevant and thus can not be simply omitted.

*Proof idea.* As a first step towards a polynomial bound on the witness of the path from $s$ to $t$, we will modify the notion of transition. Intuitively speaking, our aim is to consider exclusively relevant symbol occurrences.

By a *subword* we mean any subsequence of a given word. For instance, $aaccbc$ is a subword of $aacabbcbcbc$. Further, by a subtransition of $\mathrm{X} \longrightarrow \alpha_1 \ldots \alpha_k$ we mean any $\mathrm{X} \longrightarrow \beta_1 \ldots \beta_k$ such that the following conditions hold:

- *subword*: $\beta_i$ is a subword of $\alpha_i$, for all $i \in \{1 \ldots k\}$; and
- *nonemptiness*: $\beta_1 \ldots \beta_k \neq \varepsilon$, i.e., at least one of words $\beta_i$ is nonempty.

Note that relying on the notion of relevance one easily deduces that whenever there is a sequence of transitions from $s$ to $t$, then there is also sequence of *subtransitions*. Indeed, it is sufficient to remove irrelevant symbol occurrences in all transitions along the path from $s$ to $t$.

Clearly, the converse implication is not true in general. For instance, if we add symbols $\mathrm{X}_0$, A and the transition $\mathrm{X}_0 \longrightarrow \mathrm{X}_1\mathrm{A}$ to the example (9), there is a sequence of subtransitions from the configuration $\mathrm{X}_0$ to $\mathrm{X}_n$. Our aim now it to modify the notion of subtransition in such a way that the converse implication does hold as well, i.e., that existence of a sequence of subtransitions implies existence of a sequence of transitions. This requires certain amount of boring book-keeping, as defined in detail below.

*Marked subtransitions.* We will need an additional copy of every stack alphabet $\mathsf{A}_i$, denoted by $\bar{\mathsf{A}}_i$, for $i = 1 \ldots k$. Thus for every $a \in \mathsf{A}_i$ there is a corresponding marked symbol $\bar{a} \in \bar{\mathsf{A}}_i$. Formally, let the $i$th stack alphabet be $\mathsf{A}_i \cup \bar{\mathsf{A}}_i$.

A *marked subword* of a word $w \in \mathsf{A}_i^*$ is any word in $(\mathsf{A}_i \cup \bar{\mathsf{A}}_i)^*$ that may be obtained from $w$ by the following *marking procedure*:

- color arbitrary occurrences in $w$ (the idea is to color irrelevant symbol occurrences),
- mark every occurrence that is followed by any colored occurrence,
- and finally remove colored occurrences.

For instance, according to the coloring $a\mathbf{a}\mathbf{c}a\mathbf{b}\mathbf{b}c\mathbf{b}cbc$, a marked subword of $aacabbcbcbc$ is $\bar{a}\bar{a}\bar{c}cbc$.

Recall that a word $w \in \mathsf{A}_i^*$ represents a content of the $i$th stack, with the left-most symbol being the top-most. Intuitively, the idea behind the notion of marked subword is to keep track of removed occurrences that are covered by other symbols on the stack.

A notion of *marked subtransition* is a natural adaptation of the notion of subtransition. Compared to subtransitions, there are two differences: 'subword' is replaced with 'marked subword'; and whenever the left-side symbol is marked, then it may only put marked symbols on its stack. Formally, a marked subtransition of $\mathrm{X} \longrightarrow \alpha_1 \dots \alpha_k$ is any $\mathrm{X} \longrightarrow \beta_1 \dots \beta_k$ such that the following conditions hold:

- *subword*: $\beta_i$ is a marked subword of $\alpha_i$, for all $i \in \{1 \dots k\}$;
- *nonemptiness*: $\beta_1 \dots \beta_k \neq \varepsilon$, i.e., at least one of words $\beta_i$ is nonempty; and
- *marking inheritance*: if $X \in \bar{\mathsf{A}}_i$ is marked then all symbols in $\beta_i$ are marked.

Note that there are exponentially many different marked subtransitions, but each one is of polynomial size. Finally, note that every subtransition is obtained from some transition by the marking procedure as above, applied to every stack separately.

By the nonemptiness assumption on marked subtransitions we obtain a simple but crucial observation:

**Lemma 6.** *Along a sequence of marked subtransitions, the size of configuration can not decrease.*

A *marked subconfiguration* of a configuration $\langle \alpha_1, \dots, \alpha_k \rangle$ is any tuple $\langle \beta_1, \dots, \beta_k \rangle$ such that $\beta_i$ is a marked subword of $\alpha_i$ for all $i \in \{1 \dots k\}$.

**Lemma 7.** *For two configurations $s$ and $t$, the following conditions are equivalent:*

*(1) there is a sequence of transitions from $s$ to $t$,*
*(2) there is a sequence of marked subtransitions from $u$ to $t$, for some marked subconfiguration $u$ of $s$.*

*Proof.* The implication from (1) to (2) follows immediately. The sequence of marked subtransitions is obtained by application of the marking procedure to all transitions. For every transition, color in the marking procedure precisely those symbol occurrences that are irrelevant.

Now we show the implication from (2) to (1). The proof uses strong normedness.

Assume a sequence $\pi$ of marked subtransitions from $u$ to $t$, for some marked subconfiguration $u$ of $s$. Recall that each subtransition in $\pi$ has its original transition of $\mathcal{A}$. We claim that there is a sequence of transitions from $s$ to $t$, that contains the original transitions of all the marked subtransitions appearing in $\pi$, and *canceling sequences*

$$q\,\mathrm{X} \longrightarrow \dots \longrightarrow \langle q, \varepsilon, \dots, \varepsilon \rangle \tag{10}$$

for some stack symbols $X$, existing due to strong normedness assumption.

The sequence of transitions from $s$ to $t$ is constructed by reversing the marking procedure. For the ease of presentation, beside letters from $\mathsf{A}_i$, we will also use colored letters.

Start with the configuration $s$, and choose any coloring of symbol occurrences in $s$ that induces $u$ as the outcome of the marking procedure. Then consecutively apply the following rule:

- If the top-most symbol X on some stack is colored, apply a canceling sequence for X.
- Otherwise, apply the original transition of the next subtransition from $\pi$, using again some coloring that could have been used in the marking procedure.

For correctness, we need to show that all colored occurrences of symbols are eventually canceled out, as this guarantees that the final configuration is precisely $t$.

Let's inspect $\pi$. As no symbol in $t$ is marked, every marked symbol occurrence eventually disappears as a result of firing of some subtransition. Recall that marking of a symbol $\bar{\mathrm{X}}$ disappears only if the subtransition pushes nothing on the stack of $\bar{X}$. As a consequence, every colored symbol occurrence will eventually appear on the top of its stack. Thus the canceling sequence for X will be eventually applied. □

**Lemma 8.** *For two configurations $u$ and $v$, if there is a sequence of marked subtransitions from $u$ to $v$, then there is such a sequence of polynomial length wrt. the sizes of $u$, $v$ and $\mathcal{A}$.*

This is the last lemma needed for NP-membership. Its proof is omitted.

*Decision procedure.* Now we drop the stateless assumption. Note that the notion of marked subconfiguration and marked subtransition may be easily adapted to transitions that change state. We do not impose however the nonemptiness condition on transitions that change state, which is in accordance with the intuition that irrelevant symbol occurrences are removed in the marking procedure. Using Lemmas 6, 7 and 8 we will define the nondeterministic decision procedure for strongly normed weak MPDAs.

Let the two given configurations $s$ and $t$ have control states $q$ and $p$, respectively. In the first step, the algorithm guesses a number of marked subconfigurations $t_1 \dots t_{n-1}$, where $n$ is not greater than the number of states of $\mathcal{A}$, and marked subtransitions that change state:

$$t_1 \longrightarrow s_1 \qquad t_2 \longrightarrow s_2 \qquad \dots \qquad t_{n-1} \longrightarrow s_{n-1}$$

such that $s_i$ and $t_{i+1}$ have the same control states for $i \in \{0 \dots n-1\}$. For convenience, we write $s_0$ instead of $s$ and $t_n$ instead of $t$. In particular, we assume that the control state of $t_1$ is $q$, and the control state of $s_{n-1}$ is $p$. Relying on Lemma 6, it is sufficient to consider configurations of sizes satisfying the following inequalities:

$$\text{size}(s_i) \leq \text{size}(t_{i+1}) \qquad \text{for } i \in \{1 \dots n-1\}. \tag{11}$$

In the second phase, the algorithm guesses, for $i \in \{0 \dots n-1\}$, a sequence of subtransitions from $s_i$ to $t_{i+1}$ of length bounded by polynomial derived from the proof of Lemma 8; and checks that the respective sequences of subtransitions lead from $s_i$ to $t_{i+1}$, as required by Lemma 7. □

*Acknowledgements.* We are grateful to anonymous reviewers for careful reading and many valuable comments.

## References

1. M. Faouzi Atig. From multi to single stack automata. In *CONCUR*, pages 117–131, 2010.
2. M. Faouzi Atig and A. Bouajjani. On the reachability problem for dynamic networks of concurrent pushdown systems. In *RP*, pages 1–2, 2009.
3. M. Faouzi Atig, A. Bouajjani, and S. Qadeer. Context-bounded analysis for concurrent programs with dynamic creation of threads. *Logical Methods in Computer Science*, 7(4), 2011.
4. A. Bouajjani and M. Emmi. Analysis of recursively parallel programs. In *POPL*, pages 203–214, 2012.
5. A. Bouajjani, J. Esparza, and O. Maler. Reachability analysis of pushdown automata: Application to model-checking. In *CONCUR*, pages 135–150, 1997.
6. A. Bouajjani, J. Esparza, S. Schwoon, and J. Strejcek. Reachability analysis of multithreaded software with asynchronous communication. In *Software Verification: Infinite-State Model Checking and Static Program Analysis*, 2006.
7. A. Bouajjani, M. Müller-Olm, and T. Touili. Regular symbolic analysis of dynamic networks of pushdown systems. In *CONCUR*, pages 473–487, 2005.
8. O. Burkart, D. Caucal, F Moller, and B. Steffen. Verification of infinite structures. In *Handbook of Process Algebra*, pages 545–623. Elevier, 2001.
9. R. Chadha, P. Madhusudan, and M. Viswanathan. Reachability under contextual locking. In *TACAS*, pages 437–450, 2012.
10. S. Christensen. *Decidability and Decomposition in process algebras*. PhD thesis, Dept. of Computer Science, University of Edinburgh, UK, 1993.
11. W. Czerwiński, S. Fröschle, and S. Lasota. Partially-commutative context-free processes. In *Proc. CONCUR'09*, volume 5710 of *LNCS*, pages 259–273. Springer-Verlag, 2009.
12. W. Czerwiński, S. Fröschle, and S. Lasota. Partially-commutative context-free processes: expressibility and tractability. *Information and Computation*, 209:782–798, 2011.
13. W. Czerwiński and S. Lasota. Partially-commutative context-free languages. Submitted., 2012.
14. V. Diekert and G. Rozenberg. *The book of traces*. World Scientific, 1995.
15. J. Esparza. Petri nets, commutative context-free grammars, and basic parallel processes. *Fundam. Inform.*, 31(1):13–25, 1997.
16. Dung T. Huynh. Commutative grammars: The complexity of uniform word problems. *Information and Control*, 57(1):21–39, 1983.
17. V. Kahlon. Reasoning about threads with bounded lock chains. In *CONCUR*, pages 450–465, 2011.
18. V. Kahlon, F. Ivancic, and A. Gupta. Reasoning about threads communicating via locks. In *CAV*, pages 505–518, 2005.
19. M. Kretínský, V. Rehák, and J. Strejcek. Reachability is decidable for weakly extended process rewrite systems. *Inf. Comput.*, 207(6):671–680, 2009.
20. A. Lal and T. W. Reps. Reducing concurrent analysis under a context bound to sequential analysis. *Formal Methods in System Design*, 35(1):73–97, 2009.
21. D. Lugiez and Ph. Schnoebelen. The regular viewpoint on PA-processes. *Theor. Comput. Sci.*, 274(1-2):89–115, 2002.
22. R. Mayr. Process rewrite systems. *Inf. Comput.*, 156(1-2):264–286, 2000.
23. S. Qadeer and J. Rehof. Context-bounded model checking of concurrent software. In *TACAS*, pages 93–107, 2005.

# A NP membership

## A.1 Proof of Theorem 2 continued

**Proof of Lemma 8.** From now on, we will write 'subtransitions' instead of 'marked subtransitions'. As we will primarily work with subtransitions, we will use the stack alphabets $\mathsf{A}_i \cup \bar{\mathsf{A}}_i$ for $i \in \{1 \dots k\}$.

The number of subtransitions that change state is bounded by the number of states of $\mathcal{A}$, as $\mathcal{A}$ is assumed to be weak. Thus it is sufficient to prove the lemma under the assumption that the subtransitions do not change state. In other words, wlog. we may assume $\mathcal{A}$ to be stateless.

The size of the right-hand side of a marked subtransition is at least 1. Distinguish subtransitions with the size of the right-hand side equal 1, and call them *singleton subtransitions*. Clearly, the number of non-singleton subtransitions appearing in the sequence in the above claim is at most equal to the size of $v$, thus it is sufficient to concentrate on the following claim:

*Claim.* If there is a sequence of *singleton* subtransitions from a configuration $u$ to $v$ then there is such a sequence of polynomial length.

Note that the sizes of $u$ and $v$ in the above claim are necessarily the same.

Now we analyze in more detail the singleton subtransitions. Note that they have the form

$$\mathrm{X} \longrightarrow \mathrm{Y} \tag{12}$$

as the right-hand sides contain precisely one occurrence of a symbol. Consider the strongly connected components in the induced graph, with symbols as vertices, and singleton subtransitions (12) as edges.

Distinguish those singleton subtransitions (12) that stay inside one strongly connected component (in other words, such that there is a sequence of subtransitions from Y back to X) and call them *inner singleton subtransitions*. Note that the number of non-inner subtransitions that appear in the sequence of the last claim is polynomial (at most quadratic), thus the last claim is equivalent to the following one:

*Claim.* If there is a sequence of *inner singleton* subtransitions from a configuration $u$ to $v$ then there is such a sequence of polynomial length.

The rest of the proof is devoted to showing the last claim.

We start by doing a sequence of simplifying assumption without losing generality.

First, wlog. we may assume that the relation (12) is transitive, as we only care about the length of the sequence of subtransitions up to a polynomial. Thus every strongly connected component is a directed clique.

By the *type* of a clique we mean the set of stacks that are represented in the clique, i.e., the stacks that have at least one symbol in the alphabet that belongs to the clique. We may assume that there is no clique of singleton type. Indeed, otherwise the stack is essentially inactive along the path, except for the top-most symbol, and thus may be ignored in our analysis.

Further, wlog. we may also assume that every clique has *at most* one symbol belonging to every stack alphabet. Indeed, two different symbols from the same clique and the same stack alphabet can easily mutate from one into the other, when being the top-most symbol of the stack. And every symbol X may be easily made top-most by popping all symbols above X to other stacks (this can be done due to the assumption that type of cliques are not singletons).

The simplifications lead us to the following reformulation of the last claim. Let $k \geq 1$ be an integer. Assume a finite set of symbols $\mathsf{A}$, each symbol $\mathrm{X} \in \mathsf{A}$ coming with its type $\mathrm{type}(\mathrm{X}) \subseteq \{1 \dots k\}$ of cardinality at least 2. Consider the set of $k$-tuples of stacks $(\mathsf{A}^*)^k$ satisfying the following consistency condition: if X appears in the $i$th sequence then $i \in \mathrm{type}(\mathrm{X})$. Consider the following transition rules: the top-most letter of some stack may be moved to the top of some other stack, as long as the consistency is preserved.

*Claim.* If there is a sequence of transitions from some configuration $u \in (\mathsf{A}^*)^k$ to some configuration $v \in (\mathsf{A}^*)^k$, then there is such a sequence of polynomial length.

So formulated, the claim is fairly straightforward.

We will show a polynomial sequence of transitions that starts in $u$ and ends in a configuration $u'$ that has the same bottom-most symbol as $v$ on some stack. This is sufficient, as the same thing may be done for all other occurrences of symbols in $v$.

Note that we do not assume that different symbols have different types. Two symbols we call *siblings* if they have the same type and this type has two elements (thus the symbols may be placed only on two stacks).

Choose an arbitrary stack that is nonempty in $v$, say the $i$th stack, with the bottom-most symbol X. We may assume wlog. that X does not appear in $u$ on the $i$th stack (otherwise, i.e., if some occurrences of X in $u$ are on the $i$th stack, move all the occurrences of X, together with all other symbols above them, to arbitrary other stacks).

Let the $j$th stack in $u$ contain an occurrence of symbol X, for some $j \neq i$.

The sequence of steps from $u$ to $u'$ is the following:

1. Move all symbols above the chosen occurrence of X from the $j$th stack to other stacks.
2. Move all symbols from the $i$th stack to other stacks such that X is still on the top of the $j$th stack.
3. Move the chosen occurrence of symbol X to the destination $i$th stack.

Clearly step 1. may be always done. We will show that step 2. may be always done as well. We distinguish two cases.

If the symbol X is not a sibling, every other symbol may be moved, from the $i$th stack, to a stack different than the $i$th one, in such a way that after this operation $X$ will be still on the top of the $j$th stack. Indeed, assume that a symbol Y is on the top of the $i$th stack. If Y can be moved to a stack different than the $j$th one, we are done. Otherwise, Y can only occur on the $i$th and $j$th stacks. According to the assumption, X and Y are not siblings, thus there is another $k$th stack to which X can be moved. Then we proceed as follows: X is moved from $j$th to the $k$th stack, next Y is moved from the $i$th stack to the $j$th stack, and finally X is moved back from the $k$th stack to the $j$th stack.

As the second case, assume that X is a sibling. Assume that the top-most occurrence of X on the $j$th stack has been chosen. As $j \neq i$, and there is a sequence of steps from $u$ to $v$, one easily observes that no sibling of X may occur in $u$ either on the $i$th stack, or above X on the $j$th stack. Thus step 2. is clearly can be done.

This completes the proof of Lemma 8 and thus also the proof of Lemma 4, under the stateless assumption. □

**Proof of Lemma 5.** Suppose $\mathcal{A}$ is a strongly normed weak MPDA. Let $L$, $K$ be regular sets of configurations of $\mathcal{A}$ and let $\pi$ be a sequence of transitions from some configuration $s \in L$ to some configuration $t \in K$. We will demonstrate existence of configurations $s' \in L$ and $t' \in K$ such that $t'$ is of polynomial size and $s' \rightsquigarrow t'$. Importantly, we do not have to provide any bound on the size of $s'$, as the polynomial bound follows by Lemma 2.

Recall the coloring discipline used in the proof of Lemma 4. There we have used just one color; here we will use an unbounded number of different colors, as described below.

The coloring discipline will apply to all configurations appearing in $\pi$. We start by coloring all symbol occurrences in the first configuration $s$ with different colors. Then, for every transition $s_1 \longrightarrow s_2$ in $\pi$, assumed that $s_1$ has been already colored, we stipulate the following coloring rule for $s_2$ (recall that symbol occurrences in $s_2$ are divided into those corresponding to symbol occurrences in $s_1$, and fresh ones):

- If a symbol occurrence corresponds to a symbol occurrence in $s_1$, its color is the same as the color of corresponding symbol occurrence.
- Let $c$ be the color of the unique occurrence of symbol in $s_1$, say symbol X, that is involved in the transition. All fresh symbol occurrences in $s_2$ that appear on the same stack as X are colored with $c$; we say that they *inherit* the color from X. All other fresh symbol occurrences in $s_2$ are colored by new fresh colors, with the proviso that two occurrences have the same color if and only if they occur on the same stack. Thus at most $k - 1$ new fresh colors is needed for coloring fresh occurrences on other stacks, where $k$ is the number of stacks.

For any color used, and for any fixed configuration, the set of all symbol occurrences colored with that color we call *line*. Note that a line is always a subset of symbol occurrences on a single stack. Further, note that the cardinality of a line is not bounded in principle, due to the inheritance of color.

We now aim at reducing the size of the destination configuration $t'$. Roughly speaking, we will prove that $s' \rightsquigarrow t'$, for some $s' \in L$ and $t' \in K$ such that both the number of different lines in $t'$, and the cardinality of all lines in $t'$, are polynomially bounded.

For convenience, we split colors into two disjoint subsets. A color $c$ is called *active* if some symbol occurrence labeled by $c$:

- either is involved in some transition in $\pi$,
- or is a fresh symbol occurrence created by some transition in $\pi$.

Otherwise, a color is called *inactive*, i.e. occurrences of this color are present in $s$ and are not involved in any transition during the run. Likewise, the lines are also called

active or inactive. Note that inactive colors label suffixes of stacks in every configuration in $\pi$, and these suffixes do not change along $\pi$. Inactive lines are clearly singletons.

*Bounding the number of active lines.* Consider content of some stack, say the $i$th stack, in the destination configuration $t \in K$. Denote by $w \in \mathsf{A}_i^*$ its prefix colored by active colors. Every active line on the $i$th stack corresponds to an infix of $w$, and thus the coloring induces a factorization

$$w = w_1 \cdot w_2 \cdot \ldots \cdot w_m$$

determined by some $m - 1$ positions in $w$.

Recall that the language $K$ is represented by a tuple $\mathcal{B}_1 \ldots \mathcal{B}_k$ of finite automata, one automaton per stack. A run of the automaton $\mathcal{B}_i$ over the $i$th stack of $t$ labels each of the $m-1$ distinguished positions in $w$ by a state. By a standard pumping argument, there is a subword $w'$ of $w$, obtained by removing a number of lines from $w$, that contains at most as many lines as the number of states of $\mathcal{B}_i$, and such that $\mathcal{B}_i$ reaches the same state after reading $w$ and $w'$. By repeating the pumping argument for all stacks, one obtains a configuration $t'$ still belonging to $K$, that contains only a polynomial (in fact, at most quadratic) number of active lines, as required.

We only need to show that $s \rightsquigarrow t'$. In this part of the proof we will use the canceling sequences (10), available due to strong normedness. Observe that every active line that appears in $\pi$ appears as the top-most one on its stack at some configuration in $\pi$. We apply the canceling sequence for all symbol occurrences in every active line not appearing in $t'$. In order to keep the reachability $s \rightsquigarrow t'$, we apply the canceling sequence in the last configuration in $\pi$ where this line is the top-most one. Thus the disappearance of a line has no effect for the remaining lines.

*Bounding the number of inactive lines.* We repeat a pumping argument similar to the above one. Let $L$ and $K$ be represented by $\mathcal{A}_1 \ldots \mathcal{A}_k$ and $\mathcal{B}_1 \ldots \mathcal{B}_k$, respectively. Consider some $i \leq k$ and runs of automata $\mathcal{A}_i$ and $\mathcal{B}_i$ over the inactive suffix of the $i$th stack of $t'$ (or $t$). The runs label every position by a pair of states of $\mathcal{A}_i$ and $\mathcal{B}_i$, respectively. Upon a repetition of the same pair of states, a standard pumping applies. Thus the length of every inactive suffix in $v$ may be reduced to at most quadratic.

*Bounding the cardinalities of active lines.* Consider the configuration $t'$ obtained by now, and a single active line on some $i$th stack in this configuration. Let $v \in \mathsf{A}_i^*$ be the word representing the line. Thus the $i$th stack in $t'$ is of the form:

$$w_1 v\, w_2$$

for some words $w_1, w_2$. Similarly as before, we aim at applying pumping inside $v$, to reduce its length.

Let's focus on symbol occurrences in $v$ in configuration $t'$ and on the corresponding symbol occurrences in other configurations in $\pi$. Observe that all symbol occurrences in $v$ satisfy the following condition:

> *the corresponding symbol occurrence in some previous configuration was freshly created in some transition*.

Some of the above-mentioned transitions have created new lines, and some not. Among symbols in $v$, distinguish a subset containing only those occurrences that satisfy the following strengthened condition:

> *the corresponding symbol occurrence in some previous configuration was freshly created in some transition that created a new line that is represented in $t'$.*

The distinguished symbol occurrences call *non-local*, the others call *local*.

The overall number of lines in $t'$ is polynomially bounded, thus the same bound applies to the number of non-local symbol occurrences in $v$. We thus obtain:

*Claim.* There is only polynomially many non-local symbol occurrences in $v$.

Thus, it is sufficient to reduce the length of any block of local occurrences in $v$. From now on we focus on a single maximal infix $u$ of $v$ that contains only local symbol occurrences.

Those transitions in $\pi$ that involve a symbol occurrence corresponding to a symbol occurrence in $u$ use only the $i$th stack. Thus this set of transitions is essentially a stateless pushdown automaton. We will use a well-known fact:

*Claim.* For a pushdown automaton one can construct a finite automaton of polynomial size that recognizes the language of all reachable configurations of the pushdown automaton.

We will now use a standard pumping to reduce the length of $u$. As said above, a run of $\mathcal{B}_i$ over the $i$th stack of $v$ labels each position of $u$ with a state. Likewise for the automaton of the above claim. Upon a repetition of a pair of states, a standard pumping applies, as usual. This completes the proof of Lemma 5. □

### A.2 Proof of Theorem 3

A straightforward adaptation of the proof of Lemma 4 (combined with Lemma 2). Observe that irrelevant symbol occurrences must necessarily be normed, as they do not contribute to the target configuration.

## B Decidability

### B.1 Proof of Theorem 4

By virtue of Lemma 2 we may focus on the $1{\rightsquigarrow}1$ reachability only. Fix a MPDA $\mathcal{A}$ and two configurations $s$ and $t$. We will describe an algorithm to decide whether $s \rightsquigarrow_{\mathcal{A}} t$. Roughly speaking, our approach is to define a suitable well order compatible with transitions, and then apply a standard algorithm for reachability of a downward-closed (the algorithm works actually in any well-structured transition system). However, to apply the standard framework we need to introduce some additional structure in configurations. This additional structure will be intuitively described as coloring of symbols, similarly as in the proof of Lemma 4.

Recall the notion of relevant symbol occurrences, introduced in Section 4. The idea of the proof will be based on the observation that removing some irrelevant symbol occurrences has no impact on reachability of a fixed target configuration (cf. Lemma 3 from Section 4).

Fix the target configuration $t$. We will define *colored configurations* and modified transitions between colored configurations. The basic intuition is that irrelevant symbol occurrences will be colored. Note however that we don't know in advance which symbol occurrences in a given configuration $s$ are relevant and which are not, as we do not even know if $s \rightsquigarrow t$. Thus a coloring will have to be guessed.

Let $n$ be the number of states of $\mathcal{A}$ and let $m$ be the size of $t$. By a *colored configuration* we mean a configuration with some symbol occurrences colored, such that the number of uncolored symbol occurrences is smaller than $n + m$. Formally, coloring is implemented by extending the alphabet of every stack with its colored copy. We define an ordering on colored configurations: $r' \preceq r$ if $r'$ is obtained from $r$ by removing some colored symbol occurrences. (In particular, if $r' \preceq r$ then both configurations are identical, when restricted to uncolored symbols, so every single uncolored configuration is indeed a downward-closed set). As the number of uncolored occurrences is bounded, the number of blocks of colored occurrences is bounded likewise. The ordering $\preceq$ is like a Higman ordering on words, extended in the point-wise manner to blocks of colored occurrences. Thus one easily shows, using Higman's lemma:

*Claim.* The ordering $\preceq$ is a well order on colored configurations.

Now we define the transition rules for colored configurations. Consider any original transition rule $\delta$ of $\mathcal{A}$. This transition rule will give rise to a number of new transition rules that will be applicable to colored configurations. One new transition is obtained by coloring all symbols in $\delta$, i.e., both the left-hand side symbol and all the right-hand side symbol occurrences. In all other new transitions arising from $\delta$, the left-hand side symbol is kept uncolored. On the other hand, an arbitrary subset of the right-hand side symbol occurrences may be colored, under the following restriction:

> if the transition $\delta$ does not change state then at least one of right-hand side symbols must be kept uncolored.

This corresponds to the intuition that uncolored symbol occurrences correspond to relevant ones.

We have thus now two transition systems: the original transition system and the colored one. The relationship between reachability in these two systems is stated in the following claim (recall that the configuration $t$ is fixed and contains no colored symbols):

*Claim.* For any configuration $s$, $s \rightsquigarrow t$ if and only if there is some coloring $s'$ of $s$ such that $s' \rightsquigarrow t$.

Indeed, the only if direction is obtained by coloring precisely irrelevant symbol occurrences in $s$. The if direction also follows immediately, by replacing the colored transitions with their uncolored original transitions.

Basing on the above claim, the algorithm for $s \rightsquigarrow t$ simply guesses a coloring $s'$ of $s$ and then checks if $s' \rightsquigarrow t$ in the colored transition system. It thus only remains to

show that the latter problem is decidable. For this we will need a compatibility property of the colored transitions with respect to the well order:

*Claim.* For every colored configurations $r'$, $r$ and $u$, if $r' \preceq r \longrightarrow u$ then

- either there is a colored configuration $u'$ with $r' \longrightarrow u' \preceq u$,
- or $r' \preceq u$.

In other words, $\preceq$ is a variant of backward simulation with respect to $\longrightarrow$. Indeed, if the symbol occurrence involved in $r \longrightarrow u$ is uncolored, the transition may be also fired from $r'$. Otherwise, suppose that the symbol occurrence involved in $r \longrightarrow u$ is colored (recall that in this case all fresh symbol occurrences are colored). If this occurrence appears also in $r'$, it may be fired similarly as above. On the other hand, if this occurrence does not appear in $r'$, we have $r' \preceq u$, as required.

Using the last claim we easily show decidability. The algorithm explores exhaustively a portion of the tree of colored configurations reachable from $s'$, with the following termination condition. As the ordering $\preceq$ is a well order, we know that on every path eventually two colored configurations appear, say $u'$ and $u$, such that $u'$ precedes $u$ and $u' \preceq u$. Such a pair we call *domination pair*. Whenever a domination pair is found on some path, the algorithm stops lengthening this path. The well order guarantees thus that the algorithm terminates, after computing a finite tree of colored configurations. The algorithm answers 'yes' if the configuration $t$ appears in the tree.

Now we prove correctness of the algorithm. Towards contradiction, suppose $t$ is reachable from $s'$ but $t$ is not found in the tree. Consider the shortest path $\pi$ from $s'$ to $t$, and the domination pair $u' \preceq u$ on that path. Thus $u \rightsquigarrow t$. Using the compatibility condition, we deduce that $u \rightsquigarrow t$ implies $u' \rightsquigarrow t'$ for some $t' \preceq t$, along a path not longer that the path from $u$ to $t$. By the definition of $\preceq$ we obtain $t' = t$. Thus the fragment of path $\pi$ from $s'$ to $u'$, composed with the path from $u'$ to $t$ yields a path strictly shorter than $\pi$, a contradiction. □

### B.2 Proof of Theorem 5

We start by splitting the set $L$ according to the control state, into finitely many subsets, and consider these subsets separately.

*Stateless restriction.* Then we observe that it is sufficient to prove the result for stateless strongly normed MPDAs. Indeed, suppose that we already know that the backward reachability set of a regular set is an effectively computable regular set, in case of stateless strongly normed MPDAs. Moreover, for any two different states $q$ and $p$, the state-changing backward step of a regular set $L$, i.e.,

$$\{s : \text{ there is a transition } s \longrightarrow L \text{ that changes state from } q \text{ to } p\}$$

is clearly effectively computable and regular as well. Now note that there is only finitely many possible sequences of state-changes. Taking the union over all such sequences, and composing the two regularity-preservation properties along every sequence, one gets the result for weak strongly normed MPDAs.

*Restriction to fully active paths.* In order to express a second simplification, we distinguish a subset of paths. A path from a configuration $s$ to $t$ is *fully active* if some descendant of every symbol occurrence in $s$ is involved in some transition. As a second simplification, we claim that it is sufficient to show regularity of the set of all configurations $s$ that reach $L$ by a fully active path. To prove this claim, consider any monoid homomorphism $h$ that recognizes $L$. Let $M$ be the co-domain of $h$, a finite monoid. The backward reachability set of $L$ is the union

$$\bigcup_{m,n} B_m \, L_n$$

ranging over all pairs $m, n \in M$ such that $mn$ is accepting, where $L_n = h^{-1}(n)$ is the inverse image of the element $n \in M$ with respect to $h$, and $B_m$ is the backward reachability set of $L_m$ with respect to the fully active paths. Thus regularity of the sets $B_m$ clearly implies regularity of $L$.

*The proof.* Under the restriction to fully active paths, the proof is fairly easy. Let $\mathcal{A}$ be a MPDA and let $L$ be a regular set of configurations. Recall that we may assume a MPDA to be stateless and strongly normed.

The *initialized Higman ordering* over words relates $w'$ and $w$ if the words have the same first letter, and the tails of $w'$ and $w$ are related by the ordinary Higman ordering. Order the configurations by the point-wise extension of the initialized Higman ordering, denoted by $\preceq$. Observe that this order is a well order.

*Claim.* Under the restriction to fully active paths, the backward reachability set of a regular set $L$ is upward closed with respect to $\preceq$.

Indeed, assuming $s' \preceq s$ and $s' \rightsquigarrow t \in L$, one deduces $s \rightsquigarrow t$ by applying the canceling sequences. These sequences are applicable to some descendant of every symbol occurrence in $s$, as the path is fully active and the ordering is initialized.

By the above claim, the backward reachability set is determined by the minimal configurations with respect to $\preceq$. As $\preceq$ is a well order there is only finitely many minimal configurations, and thus the backward reachability set is regular.

For effectivity, we inspect the proofs of Lemmas 2 and 5 and conclude that all the minimal elements are of polynomial size with respect to the size of $L$: indeed, if $s \rightsquigarrow L$ then $s' \rightsquigarrow L$ for some $s' \preceq s$ of polynomial size. It is important to notice that the path remains fully active while decreasing the size of the source configuration. With this, the algorithm determines the minimal elements by inspecting exhaustively all configurations $s$ of polynomially bounded size, and checks for every of them if $s \rightsquigarrow L$. The restriction to only fully active paths may be imposed by a simple encoding.

The procedure may be implemented in exponential time. □

## C Undecidability

The following example shows that the backward reachability set of a relaxed regular set is not necessarily regular.

**Example 6.** The automaton uses two stacks, with alphabets $\{\mathrm{A}, \mathrm{X}, \mathrm{B}\}$ and $\{\mathrm{C}\}$. Every symbol has a disappearing rule: $\mathrm{A} \longrightarrow \varepsilon,\ \varepsilon$, and likewise for X, B and C. Additionally there is a transition rule $\mathrm{B} \longrightarrow \mathrm{C}$. Consider the relaxed regular language

$$L = \{(\mathrm{XA}^n, \mathrm{C}^n) : n \geq 0\}$$

and its backward reachability set. Denote by $K$ the subset of the backward reachability set consisting of configurations with the second stack empty. We claim that the projection of $K$ on the alphabet of the first stack is not regular. Indeed, as the only non-disappearing rule is $\mathrm{B} \longrightarrow \mathrm{C}$, configurations from $K$ have on the first stack a word of the form $w\mathrm{XA}^n$, with at least $n$ occurrences of B in $w$.

### C.1 Proof of Theorem 1

We start by considering stateless unnormed MPDAs. We reduce the problem of checking if the intersection of two context-free languages is empty.

Assume two context-free grammars in Greibach Normal Form over an input alphabet $A$. We will construct a MPDA with three stacks. Two stacks will be used to simulate derivations of the two grammars, and the other third stack will be used for storage of the input word. Formally, the alphabet of the first and second stack are the nonterminals of the two grammars, and the alphabet of the third stack contains two symbols $a_1$ and $a_2$ for every terminal symbol $a$ of the grammars. For every production

$$\mathrm{X} \to a\,\alpha \tag{13}$$

of the first grammar, there will be a transition

$$\mathrm{X} \longrightarrow \alpha,\ \varepsilon,\ a_1$$

that drops $\alpha$ on the first stack and $a$ on the third one. Likewise, for every production (13) of the second grammar, there is a transition

$$\mathrm{X} \longrightarrow \varepsilon,\ \alpha,\ a_2.$$

The initial configuration is $\langle \mathrm{X}_1, \mathrm{X}_2, \varepsilon \rangle$, where $\mathrm{X}_i$ is the initial symbol of the $i$th grammar. Finally, the regular language $L$ of target configurations constraints the first two stacks to be empty, and the third one to:

$$\{a_1 a_2 : a \in A\}^*.$$

One easily verifies that the intersection of the two grammars is nonempty if and only if some configuration from $L$ is reachable from the initial configuration.

Now we turn to weak normed MPDAs. It turns out that normedness assumption does not make reachability problem easier, in case of weak automata. Indeed, the case of stateless unnormed MPDAs easily reduces to the case of weak normed MPDAs. It is sufficient to add an additional sink state, and for every symbol X two additional transitions, to enforce normedness. The first one allows X to change state to the sink state. The other one allows X to disappear in the sink state. (This is in fact a reduction of the whole case of weak unnormed MPDAs.) □

### C.2 Proof of Theorem 6

The proof is by reduction from the Post Correspondence Problem (PCP). For a given instance of PCP, consisting of a finite set of pairs $(s_i, t_i)$ of words, $i \in \{1 \dots n\}$, we construct a stateless strongly normed MPDA $\mathcal{A}$ and a relaxed-regular set $L$ such that the PCP instance has a solution

$$s_{i_1}\, s_{i_2} \dots s_{i_k} \;=\; t_{i_1}\, t_{i_2} \dots t_{i_k} \qquad (i_j \in \{1 \dots n\} \text{ for } j \in \{1 \dots k\})$$

if and only if there exists a path from the initial configuration of $\mathcal{A}$ to $L$. Roughly speaking, a run of $\mathcal{A}$ will simply guess a PCP solution, and the target language $L$ will be used to check its correctness.

The main difficulty to overcome is the strong normedness requirement, which implies that every symbol may always disappear and not contribute to the target configuration.

*Half-solution.* We start by restricting to only the left-hand side words $s_i$ of the PCP instance. We will construct a MPDA $\mathcal{A}_1$, and a relaxed-regular language $L_1$ of configurations, so that the reachable configurations of $\mathcal{A}_1$ belonging to $L_1$ are essentially of the form (two stacks):

$$(i_1\, i_2 \dots i_k,\;\; s_{i_1}\, s_{i_2} \dots s_{i_k}). \tag{14}$$

In other words, one of stacks contains the sequence of indexes, and the other one contains the concatenation of the corresponding words $s_i$.

For technical reasons we will however need four stacks and few auxiliary nonterminal symbols. The nonterminals of $\mathcal{A}_1$ are following (superscripts indicate the stack number of every nonterminal):

- $\mathrm{G}^1$ and $\mathrm{G}^4$, used for 'guarding' symbols on their stacks, as described below;
- $i^1$ and $i^2$, for $i \in \{1 \dots n\}$, representing the $i$th word $s_i$;
- $a^3$ and $a^4$, for $a \in \Sigma$, representing alphabet letters of the PCP instance.

The initial configuration is $(\mathrm{G}^1,\ \varepsilon,\ \varepsilon,\ \mathrm{G}^4)$.

For a word $w = a_1 a_2 \dots a_m \in \Sigma^*$, we write $w^3$ to mean the word $a_1^3 a_2^3 \dots a_m^3$. Likewise for $w^4$. The transition rules of $\mathcal{A}_1$ are the following. For $i \in \{1 \dots n\}$, there are rules:

$$\mathrm{G}^1 \;\longrightarrow\; \mathrm{G}^1\, i^1,\ \varepsilon,\ s_i^3,\ \varepsilon \qquad\qquad \mathrm{G}^4 \;\longrightarrow\; \varepsilon,\ i^2,\ \varepsilon,\ \mathrm{G}^4\, s_i^4.$$

Additionally, to fulfill the strong normedness restriction we add *disappearing transition rules* of the form $X \;\longrightarrow\; \varepsilon,\ \varepsilon,\ \varepsilon,\ \varepsilon$ for all nonterminal symbols.

The target set $L_1$ is defined to contain all configurations of the form

$$(\mathrm{G}^1\, \alpha_1,\ \alpha_2,\ \alpha_3,\ \mathrm{G}^4\, \alpha_4),$$

with $\alpha_1$ almost equal to $\alpha_2$ and $\alpha_3$ almost equal to $\alpha_4$. By 'almost equal' we mean equality modulo (ignoring) the superscripts. The set $L_1$ is clearly relaxed-regular.

Let's analyze possible ways of reaching a configuration from $L_1$. Surely $\mathrm{G}^1$ and $\mathrm{G}^4$ cannot fire the disappearing transitions, because their presence is required by $L_1$. As $\mathrm{G}^1$ and $\mathrm{G}^4$ are always top-most on their stacks, all other symbols on these stacks are 'guarded' – they can not fire a disappearing transition neither. A key observation is that no symbol from other two stacks could fire a disappearing transition:

**Lemma 9.** *Every path from the initial configuration to $L$ contains no disappearing transitions.*

*Proof.* The precise proof of this fact needs a certain effort. Let us define the *weight* of a nonterminal. The intuition behind this notion is that it counts for how many letters in words $s_i$ the particular nonterminal is responsible. The definition is the following:

- $\mathrm{weight}(\mathrm{G}^1) = \mathrm{weight}(\mathrm{G}^4) = 0$
- $\mathrm{weight}(i^1) = \mathrm{weight}(i^2) = \mathrm{length}(s_i)$
- $\mathrm{weight}(a^3) = \mathrm{weight}(a^4) = 1$

Weight of a word is defined as the sum of weights of its letters. Note now that any configuration $\alpha = (\mathrm{G}_1\,\alpha_1,\ \alpha_2,\ \alpha_3,\ \mathrm{G}_4\,\alpha_4)$ reachable from $(\mathrm{G}_1,\ \varepsilon,\ \varepsilon,\ \mathrm{G}_4)$ satisfies the following inequalities:

$$\begin{aligned} \mathrm{SInv}_1(\alpha) &= \mathrm{weight}(\alpha_1) - \mathrm{weight}(\alpha_3) \geq 0 \\ \mathrm{SInv}_2(\alpha) &= \mathrm{weight}(\alpha_4) - \mathrm{weight}(\alpha_2) \geq 0. \end{aligned}$$

To see this it is enough to observe this both semi-invariants $\mathrm{SInv}_1$ and $\mathrm{SInv}_2$ equal $0$ in the initial configuration and that they never decrease due to performing a transition. In particular, every disappearing transition on the second or third stack increases one of the semi-invariants. Finally, every configuration $\alpha \in L$ satisfies the equality:

$$\mathrm{SInv}_1(\alpha) + \mathrm{SInv}_2(\alpha) = 0,$$

as $\mathrm{weight}(\alpha_1) = \mathrm{weight}(\alpha_2)$ and $\mathrm{weight}(\alpha_3) = \mathrm{weight}(\alpha_4)$. Therefore both semi-invariants are necessary equal to $0$, and thus there is no possibility for disappearing transitions to be fired. □

As a conclusion we obtain:

**Corollary 1.** *Consider any configuration in $L_1$ that is reachable from the initial configuration, and suppose that its first and forth stacks have the form:*

$$G^1\, i^1_1 \ldots i^1_k \qquad\qquad G^4\, a^4_1 \ldots a^4_m.$$

*Then it holds:*

$$s_{i_1}\ \ldots\ s_{i_k} \;=\; a_1 \ldots a_m.$$

*Complete solution.* Similarly as above, one may construct a MPDA $\mathcal{A}_2$ and a language $L_2$ for the right-hand side words $t_i$ from the PCP instance. Essentially (i.e., ignoring the technical details) the reachable configurations of $\mathcal{A}_2$ intersected with $L_2$ are (cf. (14)):

$$(i_1\, i_2 \ldots i_k,\ \ t_{i_1}\, t_{i_2} \ldots t_{i_k}).$$

Our final solution is to appropriately combine both MPDAs and both languages.

The MPDA $\mathcal{A}$ is obtained by merging $\mathcal{A}_1$ and $\mathcal{A}_2$, but the first stacks are identified. Thus $\mathcal{A}$ will have seven stacks altogether. In particular, symbols $i^1$ and $i^2$ represent now the $i$th pair $(s_i, t_i)$. All transitions are exactly as described above, however with a different numbering of stacks. The language $L$ imposes the requirements of $L_1$ and $L_2$, and additionally requires that the fourth stack of $\mathcal{A}_1$ is almost equal to the fourth stack of $\mathcal{A}_2$.

For describing the missing details we have to fix a new numbering of stacks. Let the first four stacks correspond to the stacks of $\mathcal{A}_1$, and the remaining three stacks correspond to the stacks of $\mathcal{A}_2$ different than the first one. The initial configuration of $\mathcal{A}$ is

$$(\mathrm{G}^1,\ \varepsilon,\ \varepsilon,\ \mathrm{G}^4,\ \varepsilon,\ \varepsilon,\ \mathrm{G}^7).$$

Except for the disappearing transitions, $\mathcal{A}$ has the following transition rules:

$$\begin{aligned}
\mathrm{G}^1 &\longrightarrow \mathrm{G}^1\, i^1,\ \varepsilon,\ s_i^3,\ \varepsilon,\ \varepsilon,\ t_i^6,\ \varepsilon \\
\mathrm{G}^4 &\longrightarrow \varepsilon,\ i^2,\ \varepsilon,\ \ \mathrm{G}^4\, s_i^4,\ \varepsilon,\ \varepsilon,\ \varepsilon \\
\mathrm{G}^7 &\longrightarrow \varepsilon,\ \varepsilon,\ \varepsilon,\ \varepsilon,\ i^5,\ \varepsilon,\ \ \mathrm{G}^7\, t_i^7.
\end{aligned}$$

The language $L$ contains configurations of the form:

$$(\mathrm{G}^1\, \alpha_1,\ \alpha_2,\ \alpha_3,\ \mathrm{G}^4\, \alpha_4,\ \alpha_5,\ \alpha_6,\ \mathrm{G}^7\, \alpha_7)$$

satisfying the following almost equalities:

$$\alpha_1\ =\ \alpha_2 = \alpha_5 \qquad \alpha_3\ =\ \alpha_4 = \alpha_6\ =\ \alpha_7.$$

One can easily observe that $L$ is reachable from the initial configuration if and only if the PCP instance has a solution, using exactly the same techniques as before.