Bisimilarity of Pushdown Systems is Nonelementary

Michael Benedikt[1], Stefan Göller[2], Stefan Kiefer[1], and Andrzej S. Murawski[3]

[1] University of Oxford, UK
[2] LIAFA Paris, France and Universität Bremen, Germany [*]
[3] University of Leicester, UK

Abstract. Given two pushdown systems, the bisimilarity problem asks whether they are bisimilar. While this problem is known to be decidable our main result states that it is nonelementary, improving EXPTIME-hardness, which was the previously best known lower bound for this problem. Our lower bound result holds for normed pushdown systems as well.

1 Introduction

A central problem in theoretical computer science is to decide whether two machines or systems behave equivalently. While being generally undecidable for Turing machines, a lot of research has been devoted to find subclassses of machine devices for which this problem becomes decidable. *Equivalence checking* is the problem of determining whether two systems are semantically identical.

It is well-known that even language equivalence of pushdown automata is undecidable, in fact already their universality is undecidable. On the positive side, a celebrated result due to Sénizergues states that language equivalence of deterministic pushdown automata is decidable [13]. The best known upper bound for the latter problem is a tower of exponentials [15] (see [4] for a more recent proof), while only hardness of deterministic polynomial time is known to date.

Among the numerous notions of equivalence [18] in the realm of formal verification and concurrency theory, the central one is *bisimulation equivalence* (*bisimilarity* for short), which enjoys pleasant mathematical properties. It can be seen to take the king role: There are important characterizations the bisimulation-invariant fragments of first-order logic and of monadic second-order logic in terms of modal logic [17] and of the modal μ-calculus [5], respectively. In particular, bisimilarity is a fundamental notion for process algebraic formalisms [11]. As a result, a great deal of research in the analysis of infinite-state systems (such as pushdown systems or Petri nets) has been devoted to deciding bisimilarity of two given processes, see e.g. [10] for a comprehensive overview.

A milestone result in this context has been proven by Sénizergues: Bisimilarity on pushdown systems (i.e. transition systems induced by pushdown automata without ε-transitions) is decidable [14] ; in fact, in [14] bisimilarity is proven to be decidable for the more general class of equational graphs of finite out-degree. Since pushdown systems can be viewed as an abstraction of the call-and-return behavior of a recursive program, the latter decidability result should be read as that one can decide equivalence of recursive programs in terms of their visible behavior. Concerning *decidability* the latter result can in some sense be considered as best possible since on the slightly more general classes of type -1a and type -1b rewrite systems [7] and order-two pushdown graphs [2] bisimilarity becomes undecidable.

Though being decidable, Sénizergues' algorithm for deciding bisimilarity of pushdown systems consists of two semi-decision procedures and in fact no complexity-theoretic upper bound is known for this problem to date. On the other hand, the best known lower bound for this problem is EXPTIME shown by Kučera and Mayr[9]. In [8] EXPTIME-hardness has been even be established even for the subclass basic process algebras, for which a 2EXPTIME upper bound is known [3]. Such complexity gaps are typical in the context of infinite-state systems.

In fact, in case decidability is known, the precise *computational complexity* status of bisimilarity on infinite-state systems is known only for few classes, let us mention basic parallel processes

---

(communication-free Petri nets) [6] and one-counter systems (the transition systems induced by pushdown automata over a singleton stack alphabet) [1].

Our contribution. The main result of this paper states that bisimilarity of (systems induced by) pushdown systems is nonelementary, even in the normed case. We give small descriptions of pushdown systems on which a bisimulation game is implemented that allows to push and verify encodings of nonelementarily big counters à la Stockmeyer [16]. As an important technical tool we realize deterministic verification phases in the bisimulation game by simulating non-erasing real-time transducers that are fed with the stack content. As basic gadgets, we use the well-established technique of Defender's Forcing [7]. We are optimistic that our technique gives new insights for potential further lower bounds for bisimilarity of PA processes, regularity for pushdown systems, and weak bisimilarity of basic process algebras.

Organisation.

In Section 2 we introduce preliminaries. In Section 3 we recall basics on transductions, introduce useful abbreviations for pushdown rules and recall Defender's forcing. Section 4 consists of our nonelementary lower bound proof for bisimilarity of pushdown systems.

2 Preliminaries

By $\mathbb{N} \stackrel{\text{def}}{=} \{0, 1, \ldots\}$ we denote the set of *non-negative integers*. For $n, m \in \mathbb{N}$ with we write $[n, m]$ for $\{n, n+1, \ldots, m\}$; in particular note that $[n, m] = \emptyset$ if $n > m$.

A *labelled transition system (LTS)* is a tuple $\mathcal{S} = (S, \mathcal{A}ct, \{\stackrel{a}{\to} \mid a \in \mathcal{A}ct\})$, where S is a set of *configurations*, $\mathcal{A}ct$ is a finite set of *action labels*, and $\stackrel{a}{\to} \subseteq S \times S$ is a *transition relation* for each $a \in \mathcal{A}ct$. We say that a state $s \in S$ is a *deadlock* if there is no $t \in S$ and no $a \in \mathcal{A}ct$ such that $s \stackrel{a}{\to} t$. A binary relation $R \subseteq S \times S$ is a *bisimulation* if for each $(s, s') \in R$ and each $a \in \mathcal{A}ct$, we have: (1) if $s \stackrel{a}{\to} t$, then there is some $s' \stackrel{a}{\to} t'$ with $(t, t') \in R$ and, conversely, (2) if $s' \stackrel{a}{\to} t'$, then there is some $s \stackrel{a}{\to} t$ with $(t, t') \in R$. We write $s \sim t$ is there is some bisimulation R with $(s, t) \in R$. Although not explicitly used in this paper, it is sometimes convenient to view bisimilarity as a game between Attacker and Defender. In every round of the game, there is a pebble placed on a unique state in each transition system. Attacker then chooses one transition system and moves the pebble from the pebbled state to one of its successors by an action $\stackrel{a}{\to}$, where a is some action label. Defender must imitate this by moving the pebbled state from the other system to one of its successors by the same action $\stackrel{a}{\to}$. If one player cannot move, then the other player wins. Defender wins every infinite game. Two states s and t are bisimilar if and only if Defender has a winning strategy on the game with initial pebble configuration (s, t).

A *pushdown automaton (PDA)* is a tuple $\mathcal{P} = (Q, \Gamma, \mathcal{A}ct, \hookrightarrow)$, where Q is a finite set of *control states*, Γ is a finite set of *stack symbols*, $\mathcal{A}ct$ is a finite set of *actions*, and $\hookrightarrow \subseteq (Q \times \{\varepsilon\} \times \mathcal{A}ct \times Q \times \{\varepsilon\}) \cup (Q \times \{\varepsilon\} \times \mathcal{A}ct \times Q \times \Gamma) \cup (Q \times \Gamma \times \mathcal{A}ct \times Q \times \{\varepsilon\})$ is a finite set of *internal rules*, *push rules*, and *pop rules*, respectively. The *size* of $\mathcal{P}$ is defined as $|\mathcal{P}| \stackrel{\text{def}}{=} |\Gamma| + |\mathcal{A}ct| + |\hookrightarrow|$. We write $qv \stackrel{a}{\hookrightarrow} q'w$ to mean $(q, v, a, q', w) \in \hookrightarrow$. Such a PDA $\mathcal{P}$ induces an LTS $\mathcal{S}(\mathcal{P}) \stackrel{\text{def}}{=} (Q \times \Gamma^*, \mathcal{A}ct, \{\stackrel{a}{\to} \mid a \in \mathcal{A}ct\})$, where $\stackrel{a}{\to} \stackrel{\text{def}}{=} \bigcup_{x \in \Gamma^*}\{(qvx, q'wx) \mid qv \stackrel{a}{\hookrightarrow} q'w\}$ for each $a \in \mathcal{A}ct$ We will abbreviate each configuration (q, w) in $\mathcal{S}(\mathcal{P})$ by qw; in particular the configuration (q, ϵ) will be denoted by just q.

Given a PDA $\mathcal{P} = (Q, \Gamma, \mathcal{A}ct, \hookrightarrow)$, $q_1, q_2 \in Q$ and $w_1, w_2 \in \Gamma^*$ the *PDA bisimilarity problem* asks whether $q_1 w_1 \sim q_2 w_2$ holds in $\mathcal{S}(\mathcal{P})$. In this paper we prove the following theorem:

Theorem 1. *PDA bisimilarity is nonelementary.*

3 Techniques

3.1 Large Counters

For each $\ell, n \geq 0$ we define $\mathsf{Tower}(\ell, n)$ inductively as $\mathsf{Tower}(0, n) \stackrel{\text{def}}{=} n$ and $\mathsf{Tower}(\ell + 1, n) \stackrel{\text{def}}{=} 2^{\mathsf{Tower}(\ell, n)}$. Let $\Omega_\ell \stackrel{\text{def}}{=} \{0_\ell, 1_\ell\}$ be alphabets whose letters have values: $val(0_\ell) = 0$ and $val(1_\ell) = 1$.

A $(0, n)$-*counter* is a word from Ω_0^n. The *value* $val(c)$ of a $(0, n)$-counter $c = \sigma_0 \cdots \sigma_{n-1}$ is defined as $val(c) \stackrel{\text{def}}{=} \sum_{i=0}^{n-1} 2^i \cdot val(\sigma_i)$. So the set of values $val(c)$ of $(0, n)$-counters equals $[0, 2^n - 1] = [0, \mathsf{Tower}(1, n) - 1]$. An (ℓ, n)-counter with $\ell \geq 1$ is a word $c = c_0\sigma_0 c_1\sigma_1 \cdots c_m\sigma_m$, where $m = \mathsf{Tower}(\ell, n) - 1$, each c_i is an $(\ell - 1, n)$-counter with $val(c_i) = i$, and $\sigma_i \in \Omega_\ell$ for $i \in [0, m]$. We define $val(c) \stackrel{\text{def}}{=} \sum_{i=0}^m 2^i \cdot val(\sigma_i)$. Observe that $val(c) \in [0, \mathsf{Tower}(\ell + 1, n) - 1]$ and the length of each (ℓ, n)-counter is uniquely determined by ℓ and n. We call an (ℓ, n)-counter c *zero* if $val(c) = 0$, and *ones* if $val(c) = \mathsf{Tower}(\ell + 1, n) - 1$. In the following we write $\Omega_{\leq \ell}$ for $\bigcup_{i=0}^\ell \Omega_i$. When n is clear from the context, we may speak of an ℓ-*counter* to mean an (ℓ, n)-counter.

3.2 Transductions

A (real-time and non-erasing) *transducer* is a tuple $T = (Q, q_0, \Sigma, \Upsilon, \delta)$, where Q is a finite set of *states*, $q_0 \in Q$ is an *initial state*, Σ and Υ are finite *alphabets*, and $\delta : Q \times \Sigma \to Q \times \Upsilon^+$ is a transition function with output. We say that T is *letter-to-letter* if $\delta(q, a) \in Q \times \Upsilon$ for each $q \in Q$ and each $a \in \Sigma$. We inductively extend δ to the function $\delta^* : Q \times \Sigma^* \to Q \times \Upsilon^*$ as follows: for each $w \in \Sigma^*$ and $a \in \Sigma$ we set $\delta^*(q, \varepsilon) \stackrel{\text{def}}{=} (q, \varepsilon)$ and $\delta^*(q, aw) \stackrel{\text{def}}{=} (q'', uv)$ if $\delta(q, a) = (q', u)$ and $\delta^*(q', w) = (q'', v)$. We define the *transduction* $f_T : \Sigma^* \to \Upsilon^*$ of T as $f_T(w) \stackrel{\text{def}}{=} v$, whenever $\delta^*(q_0, w) = (q, v)$ for some $q \in Q$. A transduction $f_T : \Sigma^* \to \Upsilon^*$ is said to be letter-to-letter if T is. We define the *size of T* as $|T| \stackrel{\text{def}}{=} |Q| + |\Sigma| + |\Upsilon| + \sum\{|w| : q \in Q, a \in \Sigma, \delta(q, a) = (q', w)\}$.

Given two transductions $f_1 : \Sigma_1^* \to \Upsilon^*$ and $f_2 : \Sigma_2^* \to \Upsilon^*$ with $\Sigma_1 \cap \Sigma_2 = \emptyset$, we define their *shuffle* as $f_1 \| f_2 : (\Sigma_1 \cup \Sigma_2)^* \to \Upsilon^*$ inductively for each $w \in (\Sigma_1 \cup \Sigma_2)^*$ and each $a \in (\Sigma_1 \cup \Sigma_2)$ as follows: $f_1 \| f_2(\varepsilon) \stackrel{\text{def}}{=} \varepsilon$ and $f_1 \| f_2(aw) \stackrel{\text{def}}{=} f_i(a) \cdot (f_1 \| f_2(w))$ if $a \in \Sigma_i$, for each $i \in \{1, 2\}$.

We note that from two given transducers T_1, T_2 with transductions $f_{T_1} : \Sigma_1^* \to \Upsilon^*$ and $f_{T_2} : \Sigma_2^* \to \Upsilon^*$, one can compute in time $O(|T_1| \cdot |T_2|)$ a transducer T such that $f_T = f_{T_1} \| f_{T_2}$.

We note that every non-erasing homomorphism is a transduction (witnessed by a single-state transducer). Having a transducer T, we will often write T for f_T without risk of confusion. For $w \in \Upsilon^*$, we denote by $\Sigma \mapsto w$ the homomorphism $h(a) \stackrel{\text{def}}{=} w$ for each $a \in \Sigma$.

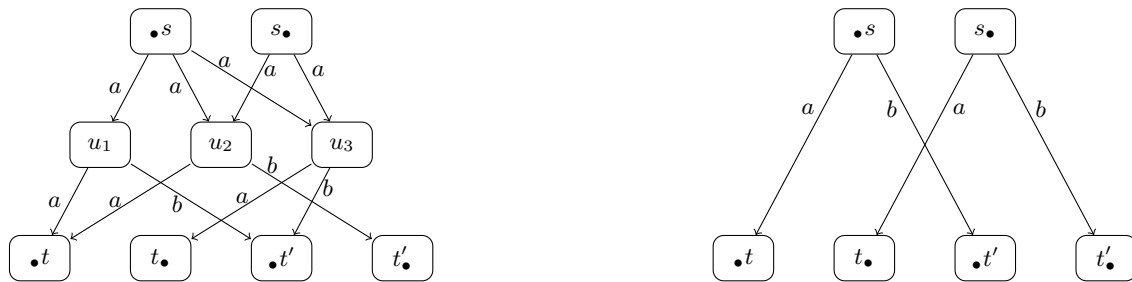### 3.3 Defender's forcing and Attacker's forcing

Fig. 1. (a) Or-gadget (Defender's forcing) (b) And-gadget (Attacker's forcing)

For our reduction we will use *Or-gadgets* ("Defender's forcing") and *And-gadgets* ("Attacker's forcing") to express logical disjunction and conjunction with bisimulation. More precisely, we have the following lemma.

Lemma 2 (see e.g. [7]). *Consider the states and transitions in Figure 1 (a) or (b) as part of an LTS. The states $\bullet s, s\bullet$ may have incoming transitions, the states $\bullet t, t\bullet, \bullet t', t'\bullet$, may have outgoing transitions (not shown). Then we have for the gadgets in Figure 1:*

(a) for the Or-gadget: $\bullet s \sim s\bullet$ if and only $\bullet t \sim t\bullet$ or $\bullet t' \sim t'\bullet$;

(b) for the And-gadget: $_\bullet s \sim s_\bullet$ if and only $_\bullet t \sim t_\bullet$ and $_\bullet t' \sim t'_\bullet$.

The lemma is easy to verify, see e.g. [7]. In terms of a Defender-Attacker game: In the Or-gadget Defender decides if the play continues in $(_\bullet t, t_\bullet)$ or in $(_\bullet t', t'_\bullet)$, whereas in the And-gadget it is Attacker who decides this.

3.4 Macro Rules

We will construct a PDA with many control states and rules. In the interest of succinctness and readability we use *macro rules* that compactly represent a set of PDA transition rules with a certain role. For the rest of the section, fix a PDA $\mathcal{P} = (Q, \Gamma, \mathcal{A}ct, \hookrightarrow)$ with $a, b \in \mathcal{A}ct$.

Macro rules with one state on the left-hand side. For $p, q \in Q$, $\sigma \in \Gamma$ and $a_1 \cdots a_\ell \in \mathcal{A}ct^\ell$ with $\ell \geq 1$ we write

$$p\sigma \xrightarrow{a_1 \cdots a_\ell} q$$

to denote that there are $p_1, \ldots, p_\ell \in Q$ with $p\sigma \xrightarrow{a_1} p_1 \xrightarrow{a_2} p_2 \cdots \xrightarrow{a_\ell} p_\ell = q$, and there are no other rules with $p\sigma$ on the left-hand side and no other rules involving $p_1, \ldots, p_{\ell-1}$.

For $p, q \in Q$ and a regular language $L \subseteq \Gamma^*$ and a transduction $T : \Gamma^* \to \mathcal{A}ct^*$ we write

$$pL \xrightarrow{T} q$$

to denote that $\mathcal{P}$ contains control states and rules described below. These rules make sure that when $\mathcal{P}$ is in a configuration py for $y \in \Gamma^*$, then the shortest prefix w of y with $w \in L$ will be popped, and $\#T(w)\#$ will be read (where $\# \in \mathcal{A}ct$ is a special action symbol), and the control state will be changed to q; if y does not have a prefix w with $w \in L$, then y will be popped, and $\#T(y)$ will be output. This behaviour is the result of a product construction between the minimal deterministic finite automaton (DFA) accepting L and the transducer T. More precisely, let $A = (Q_A, q_0^A, \Gamma, F_A, \delta_A)$ be the minimal DFA that accepts L, where Q_A is the finite set of states, $q_0^A \in Q_A$ is the initial state, $F_A \subseteq Q_A$ is the set of final states, $\delta_A : Q_A \times \Gamma \to Q_A$ is the transition function. Assume $T = (Q_T, q_0^T, \Gamma, \mathcal{A}ct, \delta_T)$. Then $\mathcal{P}$ contains the control states $Q_A \times Q_T$ and the following rules:

- $p \xrightarrow{\#} (q_0^A, q_0^T)$;
- $(q^A, q^T) \xrightarrow{\#} q$ for each $q^A \in F_A$ and each $q^T \in Q_T$;
- for each $\sigma \in \Gamma$, each $q^A \in Q_A \setminus F_A$ and each $q^T \in Q_T$, where $\delta_T(q^T, \sigma) = (r^T, w)$, we have the (macro) rule $(q^A, q^T)\sigma \xrightarrow{w} (\delta_A(q^A, \sigma), r^T)$.

There are no other rules with p on the left-hand side, and no other rules involving $Q_A \times Q_T$.

If $pL \xrightarrow{T} q$ and $w \in L$ but no proper prefix of w is in L, then we have in the LTS $\mathcal{S}(\mathcal{P})$

$$\text{for all } x \in \Gamma^* : \quad pwx \xrightarrow{\#} s_0 \xrightarrow{a_1} s_1 \xrightarrow{a_2} \ldots \xrightarrow{a_\ell} s_\ell \xrightarrow{\#} qx \,, \tag{1}$$

where the path is deterministic, $T(w) = a_1 \cdots a_\ell$, and $s_0, \ldots, s_\ell$ are configurations of $\mathcal{P}$, i.e., states in $\mathcal{S}(\mathcal{P})$.

We will need the following lemma, which shows how to compare two counters in terms of their images of two given transducers. For the statement of the lemma, recall the concept of counters and the alphabets Ω_ℓ from Section 3.1 and recall for each alphabet Ω and each word w we denote by $\Omega \mapsto w$ the homomorphism that maps every element from Ω to w.

Lemma 3. *Let $T_1, T_2 : \Omega_{\leq\ell+1}^* \to \mathcal{A}ct^*$ be letter-to-letter transducers for some $\ell \geq 0$. Let $\mathcal{P} = (Q, \Gamma, \mathcal{A}ct, \hookrightarrow)$ be a PDA with $\{_\bullet p, p_\bullet, _\bullet q, q_\bullet, _\bullet r, r_\bullet\} \subseteq Q$, $\Omega_{\leq\ell+1} \subseteq \Gamma$ and the following macro*

4

rules:

$$\bullet q \left(\Omega^*_{\leq \ell-1} \cdot \Omega_\ell\right)^* \cdot \Omega_{\ell+1} \xrightarrow{T_1} \bullet r \qquad\qquad apply\ T_1$$

$$q_\bullet \left(\Omega^*_{\leq \ell-1} \cdot \Omega_\ell\right)^* \cdot \Omega_{\ell+1} \xrightarrow{T_2} r_\bullet \qquad\qquad apply\ T_2$$

$$\bullet p\ \Omega^*_{\leq \ell} \cdot \Omega_{\ell+1} \cdot \Omega^*_{\leq \ell} \cdot \Omega_{\ell+1} \xrightarrow{\Gamma \mapsto a} \bullet q \qquad\qquad \begin{array}{l} pop\ two\ \ell\text{-}counters \\ and\ two\ \Omega_{\ell+1}\text{-}symbols \end{array}$$

$$p_\bullet\ \Omega^*_{\leq \ell} \cdot \Omega_{\ell+1} \xrightarrow{\Gamma \mapsto aa} q_\bullet \qquad\qquad \begin{array}{l} pop\ one\ \ell\text{-}counter \\ and\ one\ \Omega_{\ell+1}\text{-}symbol \end{array}$$

Let $\sigma_1, \sigma_2, \sigma_3 \in \Omega_{\ell+1}$, and let w_1, w_2, w_3 be ℓ-counters.

(a) Assume $x_1, x_2 \in \Gamma^*$ such that $\bullet r x_1 \sim r_\bullet x_2$. Then

$$\bullet q w_1 \sigma_1 x_1 \sim q_\bullet w_2 \sigma_2 x_2 \iff T_1(w_1 \sigma_1) = T_2(w_2 \sigma_2).$$

(b) Assume $x \in \Gamma^*$ such that $\bullet r x \sim r_\bullet w_1 \sigma_1 x$. Then

$$\bullet p w_3 \sigma_3 w_2 \sigma_2 w_1 \sigma_1 x \sim p_\bullet w_3 \sigma_3 w_2 \sigma_2 w_1 \sigma_1 x \iff T_1(w_1 \sigma_1) = T_2(w_2 \sigma_2).$$

Proof. Part (a) is immediate from the definitions. For part (b) we have:

$$\bullet p w_3 \sigma_3 w_2 \sigma_2 w_1 \sigma_1 x \sim p_\bullet w_3 \sigma_3 w_2 \sigma_2 w_1 \sigma_1 x$$
$$\iff \quad \bullet q w_1 \sigma_1 x \sim q_\bullet w_2 \sigma_2 w_1 \sigma_1 x \qquad\qquad \text{by the first two rules}$$
$$\iff \quad T_1(w_1 \sigma_1) = T_2(w_2 \sigma_2) \qquad\qquad \text{by part (a)}$$

$\square$

Macro rules with a state pair on the left-hand side. In the following we assume that control states, i.e., the elements of Q, are of the form $\bullet q$ and $q_\bullet$. By $\overrightarrow{q}$ we refer to the *state pair* $(\bullet q, q_\bullet) \in Q^2$. Given $w \in \Gamma^*$, we write $\sim \overrightarrow{q} w$ to denote that $\bullet q w \sim q_\bullet w$.

For $\sigma_1 \cdots \sigma_\ell \in \Gamma^\ell$ with $\ell \geq 0$ we write

$$\overrightarrow{q} \circ\!\!\rightarrow \overrightarrow{r} \sigma_1 \sigma_2 \cdots \sigma_\ell$$

to denote that there are state pairs $\overrightarrow{q_0}, \ldots, \overrightarrow{q_\ell}$ with $\overrightarrow{q_\ell} = \overrightarrow{q}$ and

$$\bullet q_\ell \xrightarrow{a} \bullet q_{\ell-1} \sigma_\ell, \ \ldots, \ \bullet q_1 \xrightarrow{a} \bullet q_0 \sigma_1, \ \bullet q_0 \xrightarrow{a} \bullet r \quad \text{and}$$

$$q_{\ell\bullet} \xrightarrow{a} q_{\ell-1\bullet} \sigma_\ell, \ \ldots, \ q_{1\bullet} \xrightarrow{a} q_{0\bullet} \sigma_1, \ q_{0\bullet} \xrightarrow{a} r_\bullet,$$

and there are no other rules with $\bullet q$ or $q_\bullet$ on the left-hand side and no other rules involving $\overrightarrow{q_0}, \ldots, \overrightarrow{q_{\ell-1}}$. With this macro rule we have

$$\text{for all } x \in \Gamma^* : \quad \sim \overrightarrow{q} x \iff \sim \overrightarrow{r} \sigma_1 \sigma_2 \cdots \sigma_\ell x. \tag{2}$$

For $\sigma_1, \sigma_2 \in \Gamma \cup \{\varepsilon\}$ we write

$$\overrightarrow{q} \xrightarrow{Def} \{\overrightarrow{r_1} \sigma_1, \ \overrightarrow{r_2} \sigma_2\}$$

to denote that Defender's forcing is implemented as described in Lemma 2 (a); i.e., in terms of Figure 1 (a) we have the state correspondences $\bullet s = \bullet q$ and $s_\bullet = q_\bullet$ and $\bullet t = \bullet r_1 \sigma_1$ and $t_\bullet = r_{1\bullet} \sigma_1$ and $\bullet t' = \bullet r_2 \sigma_2$ and $t'_\bullet = r_{2\bullet} \sigma_2$, the internal rules $\bullet q \xrightarrow{a} u_1, \ \ldots, \ q_\bullet \xrightarrow{a} u_3$ and finally the push rules $u_1 \xrightarrow{a} \bullet r_1 \sigma_1, \ \ldots, \ u_3 \xrightarrow{b} \bullet r_2 \sigma_2$, as prescribed by Figure 1 (a). Intuitively, in a Defender-Attacker game, when the play is in a configuration $(\bullet q x, q_\bullet x)$ for $x \in \Gamma^*$, then Defender chooses whether the

game will be in $(\bullet r_1\sigma_1 x, r_1 \bullet \sigma_1 x)$ or in $(\bullet r_2\sigma_2 x, r_{2,\bullet}\sigma_2 x)$. In other words, we have $\sim \overrightarrow{q} x$ iff $\sim\overrightarrow{r_1}\sigma_1 x$ or $\sim\overrightarrow{r_2}\sigma_2 x$. We generalise this notation to sets: for $\{w_1, \ldots, w_\ell\} \subseteq \Gamma^*$ we also write

$$\overrightarrow{q} \overset{Def}{\circ\!\!\longrightarrow} \{\overrightarrow{r_1}w_1,\ \ldots,\ \overrightarrow{r_\ell}w_\ell\}$$

to denote that a sequence of Or-gadgets (Figure 1 (a)) is used to achieve

$$\text{for all } x \in \Gamma^*: \quad \sim\overrightarrow{q}\,x \quad \Longleftrightarrow \quad \bigvee_{i=1}^{\ell} \sim\overrightarrow{r_i}w_i x\,. \tag{3}$$

Similarly we write

$$\overrightarrow{q} \overset{Att}{\circ\!\!\longrightarrow} \{\overrightarrow{r_1}w_1,\ \ldots,\ \overrightarrow{r_\ell}w_\ell\}$$

to denote that And-gadgets (Attacker's forcing, Figure 1 (b)) are used to achieve

$$\text{for all } x \in \Gamma^*: \quad \sim\overrightarrow{q}\,x \iff \bigwedge_{i=1}^{\ell} \sim\overrightarrow{r_i}w_i x\,. \tag{4}$$

4 The Construction

We prove Theorem 1 by showing that PDA bisimilarity is k-EXPSPACE-hard for all $k \geq 1$.

As the first step of our reduction we consider a problem on letter-to-letter transducers. A *transducer machine* is a triple $\mathcal{T} = (\ell, T_1, T_2)$, where $\ell \geq 1$, and $T_1, T_2 : \{0,1\}^* \to \Upsilon^*$ are letter-to-letter transducers. Given a transducer machine $\mathcal{T}$ we call $z \in \{0,1\}^\ell$ a *dead end* if there is no $z' \in \{0,1\}^\ell$ with $T_1(z) = T_2(z')$. We say that $\mathcal{T}$ is *deterministically terminating* if there are $t \in \mathbb{N}$ and words $z_0, \ldots, z_t \in \{0,1\}^\ell$ such that

- $z_0 = 1^\ell$,
- for each z_i there is at most one $z' \in \{0,1\}^\ell$ with $T_1(z_i) = T_2(z')$,
- $T_1(z_i) = T_2(z_{i+1})$ holds for all $i \in [0, t-1]$, and
- z_t is a dead end.

If $\mathcal{T}$ is deterministically terminating we define $\mathsf{last}(\mathcal{T}) \overset{def}{=} z_t$. The first step of our reduction is applying the algorithm of the following proposition.

Proposition 4. *For each $k \geq 1$ there exists a k-EXPSPACE-complete language $L \subseteq \Sigma^*$ such that the following is computable in polynomial time:*
INPUT: $w \in \Sigma^n$.
OUTPUT: Transducers $T_1, T_2 : \{0,1\}^* \to \Upsilon^*$, where $\mathcal{T} = (\mathsf{Tower}(k,n), T_1, T_2)$ is a deterministically terminating transducer machine and $0^{\mathsf{Tower}(k,n)}$ is a dead end such that moreover

$$w \in L \quad \textit{if and only if} \quad \mathsf{last}(\mathcal{T}) = 0^{\mathsf{Tower}(k,n)}\,.$$

Proof (sketch). Let us fix some $k \geq 1$. Let us first mention how the language L can be chosen. The following claim is a simple adaption of the linear speedup theorem, we refer the reader to [12] for details.

Claim: For each $k \geq 1$ there exists a deterministic Turing machine (DTM) $\mathcal{M}$ and some $m \in \mathbb{N}$ such that the following holds:

(1) m is the sum of the number of states of $\mathcal{M}$ plus the number of tape symbols of $\mathcal{M}$.
(2) the DTM $\mathcal{M}$ is $\left\lfloor \frac{\mathsf{Tower}(k,n)}{n+m} \right\rfloor - n$ space bounded for all but finitely many $n \in \mathbb{N}$.
(3) For each $n \geq 0$ we have
    (i) the DTM $\mathcal{M}$ has a unique accepting configuration and a unique rejecting configuration that both do not have any successor configuration on each input of length n and

(ii) the unique computation of $\mathcal{M}$ on w either reaches the accepting or the rejecting configuration for each word w of length n.

(4) The acceptance problem of $\mathcal{M}$ is complete for k-EXPSPACE under polynomial time many-one reductions.

Let us fix some $k \geq 1$ and some Turing machine $\mathcal{M}$ that satisfies points (1) to (4) of the above Claim for the rest of this proof. We define $L \stackrel{\text{def}}{=} L(\mathcal{M})$. Let us assume that the input alphabet of $\mathcal{M}$ is Σ. Let us fix an input word $w \in \Sigma^n$. We can assume without loss of generality that $\mathcal{M}$ is $\left\lfloor \frac{\mathsf{Tower}(k,n)}{n+m} \right\rfloor - n$ space bounded on input w (the other finitely many cases can be dealt with explicitly in our reduction). Recall that by Point (3) of the above Claim we have that $\mathcal{M}$ has a unique accepting configuration and a unique rejecting configuration, and that all computations of $\mathcal{M}$ reach either the accepting or the rejecting configuration, and that the accepting and rejecting configurations have no successor configurations. In a first step, we modify $\mathcal{M}$ to a DTM $\mathcal{M}_w$ so that $\mathcal{M}_w$ started on the empty tape first writes w on the tape and then simulates $\mathcal{M}$ on w. We note that we can construct $\mathcal{M}_w$ in such a way that

- $\mathcal{M}_w$ can be obtained from $\mathcal{M}$ by adding at most n additional states and corresponding transitions that allow us to initially copy w onto the working tape,
- the sum of the number of states of $\mathcal{M}_w$ plus the number of tape symbols of $\mathcal{M}_w$ is $n + m$,
- $\mathcal{M}_w$ is $\left\lfloor \frac{\mathsf{Tower}(k,n)}{n+m} \right\rfloor - n + n = \left\lfloor \frac{\mathsf{Tower}(k,n)}{n+m} \right\rfloor$ space bounded and
- $w \in L(\mathcal{M})$ if and only if $\varepsilon \in L(\mathcal{M}_w)$.

Fix a binary encoding enc of configurations so that each tape symbol and each pair consisting of a state and a tape symbol of $\mathcal{M}_w$ can be (injectively) encoded by a binary string of length $n+m$. We extend this encoding to configurations of $\mathcal{M}_w$ by mapping each configuration c (injectively) to a string $enc(c) \in \{0,1\}^\ell$ where $\ell = \mathsf{Tower}(k,n)$. Moreover we assume that the initial configuration of $\mathcal{M}_w$ (with empty tape) is encoded by 1^ℓ and that the (unique) accepting configuration of $\mathcal{M}_w$ is encoded by 0^ℓ. It remains to argue that one can construct transducers $T_1, T_2 : \{0,1\}^* \to \Upsilon^*$ so that

(*) for all configurations c, c' of $\mathcal{M}_w$ we have $T_1(enc(c)) = T_2(enc(c'))$ if and only if c' is a successor configuration (i.e., the unique one) of c.

For establishing (*), the idea is to construct T_1, T_2 so that if c is a configuration of $\mathcal{M}_w$ then $T_1(enc(c))$ is an encoding of c', where c' is the successor configuration of c, and $T_2(enc(c))$ is an encoding of c. The most straightforward implementation of this idea would be to let $T_1(enc(c)) = enc(c')$ and $T_2(enc(c)) = enc(c)$. However, this cannot be easily done, if at all, loosely speaking because the read-write head of $\mathcal{M}_w$ may move in the direction "opposite" to the transducers so that the transducer T_1 would have to "guess" where the read-write head is before it actually sees it. Therefore, we construct T_1, T_2 so that their output is "delayed" by a few steps: Transducer T_1 remembers in its finite control the last few bits of the encoded tape and outputs the bits of the encoding of the successor configuration only after T_1 can be sure about them. Transducer T_2 does not compute the successor configuration, but only re-encodes the encoded configuration, and outputs the bits of the new encoding in a similarly delayed way as T_1. Since transducers need to output a single symbol per step, the transducers T_1, T_2 output a dummy symbol in the first few steps. At the end they need to output a single symbol containing the last few bits of the new encoding. As a consequence the alphabet Υ cannot be (easily) taken to be binary; thus we simply choose Υ sufficiently large for this to work. $\square$

Let us fix some $k \geq 1$ for the rest of this section and let us fix the k-EXPSPACE complete language $L \subseteq \Sigma^*$ that satisfies Proposition 4. Moreover let $w \in \Sigma^n$ be a word. Our overall goal is to compute from w in polynomial time a PDA and two of its configurations that are bisimilar if and only if $w \in L$. As an intermediate step, let us fix for the rest of this section the output (T_1, T_2) of the algorithm of Proposition 4 on input w, and let $\mathcal{T} = (\mathsf{Tower}(k,n), T_1, T_2)$. In the rest of the section we will show how to compute from T_1, T_2 and n in time polynomial in $|T_1| + |T_2| + n$ a

PDA $\mathcal{P} = (Q, \Omega, \mathcal{A}ct, \hookrightarrow)$ so that we have $\sim\overrightarrow{\langle\mathsf{start}\rangle}$ in $\mathcal{S}(\mathcal{P})$ if and only if $\mathsf{last}(\mathcal{T}) = 0^{\mathsf{Tower}(k,n)}$ holds in $\mathcal{T}$, where $_\bullet\langle\mathsf{start}\rangle$ and $\langle\mathsf{start}\rangle_\bullet$ will be control states of $\mathcal{P}$. We recall that k is a fixed constant. Let

$$B \overset{\text{def}}{=} \{\mathsf{start}, \mathsf{stop}_\ell, \mathsf{testDec}_\ell, \mathsf{testDec}_\ell^1, \mathsf{ones}_\ell, \mathsf{ones}_\ell^1, \mathsf{decOk}_\ell, \mathsf{zero}_\ell, \mathsf{zero}_\ell^1, \mathsf{dec}_\ell, \mathsf{dec}_\ell^1, \mathsf{dec}_0^{(i)},$$
$$\mathsf{fin}, \mathsf{testFin}, \mathsf{popAll}, \mathsf{next}, \mathsf{next}^1, \mathsf{tran}, \mathsf{testTran}, \mathsf{testTran}^1 \mid 0 \le \ell \le k+2,\ 1 \le i \le n\} \cup B_{impl}$$

be a set of "basic symbols" that we use to construct the control states Q. The set B_{impl} contains further (implicit) symbols that are needed to implement macro rules. In the following we regard each element of B as a single symbol (of length 1). Define $\Omega \overset{\text{def}}{=} \Omega_{\le k+1}$, $\mathcal{A}ct \overset{\text{def}}{=} \{0, 1, \#, a, b\} \uplus \Upsilon$, $Q = {}_\bullet B \cup B_\bullet$, where ${}_\bullet B \overset{\text{def}}{=} \{{}_\bullet\langle\alpha\rangle \mid \alpha \in B^*,\ 1 \le |\alpha| \le k+2\}$ and similarly $B_\bullet \overset{\text{def}}{=} \{\langle\alpha\rangle_\bullet \mid \alpha \in B^*,\ 1 \le |\alpha| \le k+2\}$. For instance, we have ${}_\bullet\langle\mathsf{dec}_{k-1}\ \mathsf{ones}_k\rangle \in Q$. We will use β to indicate an arbitrary word $\alpha \in B^*$ with $1 \le |\alpha| \le k+1$.

This section is organised as follows. In Section 4.1 we show how we can implement a bisimulation (sub-)game in $\mathcal{S}(\mathcal{P})$ that allows us to test whether two ℓ-counters have consecutive values for each $\ell \ge 0$. In Section 4.2 we show how we can implement a bisimulation (sub-)game in $\mathcal{S}(\mathcal{P})$ that allows Defender to push an ℓ-counter onto the stack for each $\ell \ge 0$. We conclude our reduction in Section 4.3.

4.1 Checking Counters for Consecutive Values

For each $\ell \in [0, k]$ we include control states $_\bullet\langle\mathsf{stop}_\ell\rangle, \langle\mathsf{stop}_\ell\rangle_\bullet$ such that for all $x \in \Omega^*$ and all $\sigma \in \Omega_{\ell+1}$ and all ℓ-counters w we have

$$_\bullet\langle\mathsf{stop}_\ell\rangle x \sim \langle\mathsf{stop}_\ell\rangle_\bullet w\sigma x \tag{5}$$

This is easily achieved, for instance by including no rules with $_\bullet\langle\mathsf{stop}_\ell\rangle$ or $\langle\mathsf{stop}_\ell\rangle_\bullet$ on the left-hand side.

We need to be able to verify whether two counters (at convenient positions) on the stack have consecutive values. To this end we include rules such that the following statement holds:

Lemma 5. *Let* $x \in \Omega^*$, *and* $\sigma_1, \sigma_2, \sigma_3 \in \Omega_{\ell+1}$, *and let* w_1, w_2, w_3 *be* ℓ*-counters. Then* $\sim\overrightarrow{\langle\mathsf{testDec}_\ell\rangle}w_3\sigma_3 w_2\sigma_2 w_1\sigma_1 x$ *iff* $val(w_1) = val(w_2) + 1$.

Let $T_\ell^{+0}, T_\ell^{+1} : (\Omega_\ell \cup \Omega_{\ell+1})^* \to \{0, 1, a, b\}^*$ be the transducers depicted in Figure 2. Transducers T_ℓ^{+0}, T_ℓ^{+1} interpret the input word over Ω_ℓ as a number in binary, with the least

Fig. 2. Transducers T_ℓ^{+0} and T_ℓ^{+1}

significant bit read first. Transducer T_ℓ^{+0} copies the number and outputs a upon reading an $\Omega_{\ell+1}$-symbol. Transducer T_ℓ^{+1} attempts to increase the number by 1 and output a upon reading an

$\Omega_{\ell+1}$-symbol, but it outputs b if the input number consisted only of 1s. If w_1, w_2 are ℓ-counters and $\sigma_1, \sigma_2 \in \Omega_{\ell+1}$, then we have:

$$(T_\ell^{+0} \| \Omega_{\leq \ell-1} \mapsto a)(w_1 \sigma_1) = (T_\ell^{+1} \| \Omega_{\leq \ell-1} \mapsto a)(w_2 \sigma_2)$$
$$\text{iff} \quad val(w_1) = val(w_2) + 1 \tag{6}$$

Transducers T_ℓ^{+0}, T_ℓ^{+1} are used in the following rules.

$$\bullet \langle \mathsf{testDec}_\ell^1 \rangle \left(\Omega_{\leq \ell-1}^* \cdot \Omega_\ell \right)^* \cdot \Omega_{\ell+1} \xrightarrow{T_\ell^{+0} \| \Omega_{\leq \ell-1} \mapsto a} \bullet \langle \mathsf{stop}_\ell \rangle$$

$$\langle \mathsf{testDec}_\ell^1 \rangle_\bullet \left(\Omega_{\leq \ell-1}^* \cdot \Omega_\ell \right)^* \cdot \Omega_{\ell+1} \xrightarrow{T_\ell^{+1} \| \Omega_{\leq \ell-1} \mapsto a} \langle \mathsf{stop}_\ell \rangle_\bullet$$

$$\bullet \langle \mathsf{testDec}_\ell \rangle \Omega_{\leq \ell}^* \cdot \Omega_{\ell+1} \cdot \Omega_{\leq \ell}^* \cdot \Omega_{\ell+1} \xrightarrow{\Omega \mapsto a} \bullet \langle \mathsf{testDec}_\ell^1 \rangle$$

$$\langle \mathsf{testDec}_\ell \rangle_\bullet \Omega_{\leq \ell}^* \cdot \Omega_{\ell+1} \xrightarrow{\Omega \mapsto aa} \langle \mathsf{testDec}_\ell^1 \rangle_\bullet$$

Proof (of Lemma 5). By (5) we can apply Lemma 3 (b). Hence we have:

$$\sim \overrightarrow{\langle \mathsf{testDec}_\ell \rangle} w_3 \sigma_3 w_2 \sigma_2 w_1 \sigma_1 x$$
$$\iff (T_\ell^{+0} \| \Omega_{\leq \ell-1} \mapsto a)(w_1 \sigma_1) = (T_\ell^{+1} \| \Omega_{\leq \ell-1} \mapsto a)(w_2 \sigma_2) \qquad \text{by Lemma 3 (b)}$$
$$\iff val(w_1) = val(w_2) + 1 \qquad\qquad\qquad\qquad\qquad\qquad \text{by (6)}$$

$\square$

4.2 Building Counters

In the lemmas below we will make statements about properties of $\mathcal{S}(\mathcal{P})$ if we include certain rules to $\mathcal{P}$. For better readability, we will state the properties before we list the rules.

Lemma 6 below demonstrates how to construct large counters. Recall that we use β to indicate an arbitrary word from B^* with length between 1 and $k+1$. We include rules such that the following holds:

Lemma 6. *Let $\ell \in [0, k]$, and $x \in \Omega^*$, and $\sigma, \tau \in \Omega_{\ell+1}$, and v, w be ℓ-counters.*

(a) *Then $\sim \overrightarrow{\langle \mathsf{ones}_\ell \ \beta \rangle} x$ iff $\sim \overrightarrow{\langle \beta \rangle} wx$, where w is the ones ℓ-counter.*

(b) *Then $\sim \overrightarrow{\langle \mathsf{decOk}_\ell \ \beta \rangle} v\sigma w\tau x$ iff $val(v) + 1 = val(w)$ and $\sim \overrightarrow{\langle \beta \rangle} v\sigma w\tau x$.*

(c) *Then $\sim \overrightarrow{\langle \mathsf{zero}_\ell \ \beta \rangle} \sigma w\tau x$ iff $val(w) = 1$ and $\sim \overrightarrow{\langle \beta \rangle} v\sigma w\tau x$, where v is the ℓ-counter with $val(v) = 0$.*

(d) *Then $\sim \overrightarrow{\langle \mathsf{dec}_\ell \ \beta \rangle} \sigma w\tau x$ iff $val(w) \neq 0$ and $\sim \overrightarrow{\langle \beta \rangle} v\sigma w\tau x$, where v is the ℓ-counter with $val(v)+1 = val(w)$.*

The following rules for the special case $\ell = 0$ are included.

$$\overrightarrow{\langle \text{ones}_0\ \beta \rangle} \multimap \overrightarrow{\langle \beta \rangle} 1_0^n \qquad\qquad \text{push a ones 0-counter}$$

$$\overrightarrow{\langle \text{decOk}_0\ \beta \rangle} \xrightarrow{Att} \Big\{ \overrightarrow{\langle \beta \rangle}, \qquad\qquad \begin{array}{l}\text{believe that the values of the}\\ \text{top two 0-counters differ by 1}\end{array}$$

$$\overrightarrow{\langle \text{testDec}_0 \rangle} 0_0^n 0_1 \Big\} \qquad \begin{array}{l}\text{OR challenge that claim by}\\ \text{invoking } \text{testDec}_0\end{array}$$

$$\overrightarrow{\langle \text{zero}_0\ \beta \rangle} \multimap \overrightarrow{\langle \text{decOk}_0\ \beta \rangle} 0_0^n \qquad \begin{array}{l}\text{push a zero 0-counter and check}\\ \text{if it is over a 0-counter with value 1}\end{array}$$

$$\overrightarrow{\langle \text{dec}_0\ \beta \rangle} \xrightarrow{Def} \Big\{ \overrightarrow{\langle \text{dec}_0^{(1)}\ \beta \rangle} 0_0, \overrightarrow{\langle \text{dec}_0^{(1)}\ \beta \rangle} 1_0 \Big\} \qquad \begin{array}{l}\text{push the first bit of the}\\ \text{decremented 0-counter}\end{array}$$

$$\forall 1 \le i < n: \overrightarrow{\langle \text{dec}_0^{(i)}\ \beta \rangle} \xrightarrow{Def} \Big\{ \overrightarrow{\langle \text{dec}_0^{(i+1)}\ \beta \rangle} 0_0, \overrightarrow{\langle \text{dec}_0^{(i+1)}\ \beta \rangle} 1_0 \Big\} \qquad \begin{array}{l}\text{push the } (i+1)^{\text{st}} \text{ bit of the}\\ \text{decremented 0-counter}\end{array}$$

$$\overrightarrow{\langle \text{dec}_0^{(n)}\ \beta \rangle} \multimap \overrightarrow{\langle \text{decOk}_0\ \beta \rangle} \qquad \begin{array}{l}\text{verify if the 0-counter has been}\\ \text{correctly decremented}\end{array}$$

The following rules are included for $1 \le \ell \le k$.

$$\overrightarrow{\langle \text{ones}_\ell\ \beta \rangle} \multimap \overrightarrow{\langle \text{ones}_{\ell-1}\ \text{ones}_\ell^1\ \beta \rangle} 1_\ell \qquad \text{push } 1_\ell \text{ and a ones } (\ell-1)\text{-counter}$$

$$\overrightarrow{\langle \text{ones}_\ell^1\ \beta \rangle} \xrightarrow{Def} \Big\{ \overrightarrow{\langle \text{dec}_{\ell-1}\ \text{ones}_\ell^1\ \beta \rangle} 1_\ell, \qquad \text{push } 1_\ell \text{ and a decremented } (\ell-1)\text{-counter}$$

$$\overrightarrow{\langle \text{zero}_{\ell-1}\ \beta \rangle} 1_\ell \Big\} \qquad \text{OR push } 1_\ell \text{ and a zero } (\ell-1)\text{-counter}$$

$$\overrightarrow{\langle \text{decOk}_\ell\ \beta \rangle} \xrightarrow{Att} \Big\{ \overrightarrow{\langle \beta \rangle}, \qquad \begin{array}{l}\text{believe that the values of the}\\ \text{top two } \ell\text{-counters differ by 1}\end{array}$$

$$\overrightarrow{\langle \text{ones}_\ell\ \text{testDec}_\ell \rangle} 0_{\ell+1} \Big\} \qquad \begin{array}{l}\text{OR challenge that claim by}\\ \text{invoking } \text{testDec}_\ell\end{array}$$

$$\overrightarrow{\langle \text{zero}_\ell\ \beta \rangle} \multimap \overrightarrow{\langle \text{ones}_{\ell-1}\ \text{zero}_\ell^1\ \beta \rangle} 0_\ell \qquad \text{push } 0_\ell \text{ and a ones } (\ell-1)\text{-counter}$$

$$\overrightarrow{\langle \text{zero}_\ell^1\ \beta \rangle} \xrightarrow{Def} \Big\{ \overrightarrow{\langle \text{dec}_{\ell-1}\ \text{zero}_\ell^1\ \beta \rangle} 0_\ell, \qquad \text{push } 0_\ell \text{ and a decr. } (\ell-1)\text{-counter}$$

$$\overrightarrow{\langle \text{zero}_{\ell-1}\ \text{decOk}_\ell\ \beta \rangle} 0_\ell \Big\} \qquad \text{OR push } 0_\ell \text{ and a zero } (\ell-1)\text{-counter}$$

$$\overrightarrow{\langle \text{dec}_\ell\ \beta \rangle} \xrightarrow{Def} \Big\{ \overrightarrow{\langle \text{ones}_{\ell-1}\ \text{dec}_\ell^1\ \beta \rangle} \sigma \mid \sigma \in \Omega_\ell \Big\} \qquad \text{push from } \Omega_\ell \text{ and a ones } (\ell-1)\text{-counter}$$

$$\overrightarrow{\langle \text{dec}_\ell^1\ \beta \rangle} \xrightarrow{Def} \Big\{ \overrightarrow{\langle \text{dec}_{\ell-1}\ \text{dec}_\ell^1\ \beta \rangle} \sigma, \qquad \text{push from } \Omega_\ell \text{ and a decr. } (\ell-1)\text{-counter}$$

$$\overrightarrow{\langle \text{zero}_{\ell-1}\ \text{decOk}_\ell\ \beta \rangle} \sigma \mid \sigma \in \Omega_\ell \Big\} \qquad \text{OR push from } \Omega_\ell \text{ and a zero } (\ell-1)\text{-counter}$$

Proof (of Lemma 6). The proof is by induction on ℓ.

Induction base.

Case (a) immediately follows from the rule for $\overrightarrow{\langle \text{ones}_0\ \beta \rangle}$.

For part (b) we have:

$$\sim\overrightarrow{\langle \mathsf{decOk}_0\ \beta\rangle}v\sigma w\tau x \quad\Longleftrightarrow\quad \sim\overrightarrow{\langle \mathsf{testDec}_0\rangle}0_0^n 0_1 v\sigma w\tau x \text{ and } \sim\overrightarrow{\langle\beta\rangle}v\sigma w\tau x \quad (\text{rule for } \overrightarrow{\langle \mathsf{decOk}_0\beta\rangle})$$

$$\Longleftrightarrow\quad val(v)+1 = val(w) \text{ and } \sim\overrightarrow{\langle\beta\rangle}v\sigma w\tau x \qquad (\text{Lemma 5})$$

For part (c) we have:

$$\sim\overrightarrow{\langle \mathsf{zero}_0\ \beta\rangle}\sigma w\tau x \quad\Longleftrightarrow\quad \sim\overrightarrow{\langle \mathsf{decOk}_0\rangle}0_0^n\sigma w\tau x \qquad\qquad (\text{rule for } \overrightarrow{\langle \mathsf{zero}_0\beta\rangle} \text{ and part (b)})$$

$$\Longleftrightarrow\quad val(0_0^n)+1 = val(w) \text{ and } \sim\overrightarrow{\langle\beta\rangle}v\sigma w\tau x \quad (\text{Lemma 5})$$

$$\Longleftrightarrow\quad val(w) = 1 \text{ and } \sim\overrightarrow{\langle\beta\rangle}v\sigma w\tau x,\text{ where}$$
$$v \text{ is the 0-counter with } val(v)=0$$

For part (d) we have:

$$\sim\overrightarrow{\langle \mathsf{dec}_0\ \beta\rangle}\sigma w\tau x \quad\Longleftrightarrow\quad \exists v \in \Omega_0^n : \sim\overrightarrow{\langle \mathsf{decOk}_0\ \beta\rangle}v\sigma w\tau x \qquad (\text{rules for } \overrightarrow{\langle \mathsf{dec}_0\ \beta\rangle} \text{ and } \overrightarrow{\langle \mathsf{dec}_0^{(i)}\ \beta\rangle})$$

$$\Longleftrightarrow\quad \exists v \in \Omega_0^n : val(v)+1 = val(w) \text{ and } \sim\overrightarrow{\langle\beta\rangle}v\sigma w\tau x \quad (\text{part (b)})$$

$$\Longleftrightarrow\quad val(w) \neq 0 \text{ and } \sim\overrightarrow{\langle\beta\rangle}v\sigma w\tau x,\text{ where } v$$
$$\text{is the } \ell\text{-counter with } val(v)+1 = val(w)$$

Induction step. In the following let $\ell \in [0, k-1]$, let $x \in \Omega^*$, let $\sigma, \tau \in \Omega_{\ell+2}$ and let v, w be $(\ell+1)$-counters. Let $m \overset{\text{def}}{=} \mathsf{Tower}(n, \ell+1) - 1$. We write c_i for the ℓ-counter with $val(c_i) = i$ for each $i \in [0, m]$.

For part (a) we obtain the following equivalences:

$$\sim\overrightarrow{\langle \mathsf{ones}_{\ell+1}\ \beta\rangle}x \quad\Longleftrightarrow\quad \sim\overrightarrow{\langle \mathsf{ones}_\ell\ \mathsf{ones}_{\ell+1}^1\ \beta\rangle}1_{\ell+1}x \qquad\qquad (\text{rule for } \overrightarrow{\langle \mathsf{ones}_{\ell+1}\ \beta\rangle})$$

$$\Longleftrightarrow\quad \sim\overrightarrow{\langle \mathsf{ones}_{\ell+1}^1\ \beta\rangle}c_m 1_{\ell+1}x \qquad\qquad (\text{ind. hyp. on (a)})$$

$$\Longleftrightarrow\quad \sim\overrightarrow{\langle \mathsf{dec}_\ell\ \mathsf{ones}_{\ell+1}^1\ \beta\rangle}1_{\ell+1}c_m 1_{\ell+1}x \qquad\qquad (\text{rule for } \overrightarrow{\langle \mathsf{ones}_{\ell+1}^1\ \beta\rangle})$$
$$\text{or } \sim\overrightarrow{\langle \mathsf{zero}_\ell\ \beta\rangle}1_{\ell+1}c_m 1_{\ell+1}x$$

$$\Longleftrightarrow\quad \sim\overrightarrow{\langle \mathsf{ones}_{\ell+1}^1\ \beta\rangle}c_{m-1}1_{\ell+1}c_m 1_{\ell+1}x \qquad\qquad (\text{ind. hyp. on (d),(c)}, m > 1)$$

$$\Longleftrightarrow\quad \cdots$$

$$\Longleftrightarrow\quad \sim\overrightarrow{\langle \mathsf{dec}_\ell\ \mathsf{ones}_{\ell+1}^1\ \beta\rangle}1_{\ell+1}c_1\cdots 1_{\ell+1}c_m 1_{\ell+1}x$$
$$\text{or } \sim\overrightarrow{\langle \mathsf{zero}_\ell\ \beta\rangle}1_{\ell+1}c_1\cdots 1_{\ell+1}c_m 1_{\ell+1}x$$

$$\Longleftrightarrow\quad \sim\overrightarrow{\langle \mathsf{ones}_{\ell+1}^1\ \beta\rangle}c_0 1_{\ell+1}\cdots c_m 1_{\ell+1}x \qquad\qquad (\text{ind. hyp. on (d)})$$
$$\text{or } \sim\overrightarrow{\langle\beta\rangle}\underbrace{c_0 1_{\ell+1}\cdots 1_{\ell+1}c_m 1_{\ell+1}}_{\text{ones } (\ell+1)\text{-counter}} x \qquad\qquad (\text{ind. hyp. on (c)})$$

Further we have

$$\sim\overrightarrow{\langle \mathsf{ones}_{\ell+1}^1\ \beta\rangle}c_0 1_{\ell+1}\cdots c_m 1_{\ell+1}x$$

$$\Longleftrightarrow\quad \sim\overrightarrow{\langle \mathsf{dec}_\ell\ \mathsf{ones}_{\ell+1}^1\ \beta\rangle}1_{\ell+1}c_0 1_{\ell+1}\cdots c_m 1_{\ell+1}x \qquad\qquad (\text{rule for } \overrightarrow{\langle \mathsf{ones}_{\ell+1}^1\ \beta\rangle})$$
$$\text{or } \sim\overrightarrow{\langle \mathsf{zero}_\ell\ \beta\rangle}1_{\ell+1}c_0 1_{\ell+1}\cdots c_m 1_{\ell+1}x$$

$$\Longleftrightarrow\quad \textit{false} \qquad\qquad (\text{ind. hyp. on (d),(c)})$$

11

Combining this with the equivalences above yields part (a) for $\ell + 1$.

For part (b) we obtain the following equivalences:

$$\sim\overrightarrow{\langle\mathsf{decOk}_{\ell+1}\ \beta\rangle}v\sigma w\tau x \quad\Longleftrightarrow\quad \sim\overrightarrow{\langle\beta\rangle}v\sigma w\tau x$$

$$\text{and } \sim\overrightarrow{\langle\mathsf{ones}_{\ell+1}\mathsf{testDec}_{\ell+1}\rangle}0_{\ell+2}v\sigma w\tau x \qquad (\text{rule for } \overrightarrow{\langle\mathsf{decOk}_{\ell+1}\ \beta\rangle})$$

$$\Longleftrightarrow\quad \sim\overrightarrow{\langle\beta\rangle}v\sigma w\tau x \text{ and } \sim\overrightarrow{\langle\mathsf{testDec}_{\ell+1}\rangle}w'0_{\ell+2}v\sigma w\tau x$$

$$\text{where } w' \text{ is the ones } (\ell+1)\text{-counter} \qquad (\text{part (a)})$$

$$\Longleftrightarrow\quad \sim\overrightarrow{\langle\beta\rangle}v\sigma w\tau x \text{ and } val(w) = val(v) + 1 \qquad (\text{Lemma 5})$$

For part (c) we obtain the following equivalences:

$$\sim\overrightarrow{\langle\mathsf{zero}_{\ell+1}\beta\rangle}\sigma w\tau x$$

$$\Longleftrightarrow\quad \sim\overrightarrow{\langle\mathsf{ones}_{\ell}\mathsf{zero}^1_{\ell+1}\ \beta\rangle}0_{\ell+1}\sigma w\tau x \qquad (\text{rule for } \overrightarrow{\langle\mathsf{zero}_{\ell+1}\ \beta\rangle})$$

$$\Longleftrightarrow\quad \sim\overrightarrow{\langle\mathsf{zero}^1_{\ell+1}\ \beta\rangle}c_m 0_{\ell+1}\sigma w\tau x \qquad (\text{ind. hyp. on (a)})$$

$$\Longleftrightarrow\quad \sim\overrightarrow{\langle\mathsf{dec}_{\ell}\mathsf{zero}^1_{\ell+1}\ \beta\rangle}0_{\ell+1}c_m 0_{\ell+1}\sigma w\tau x$$

$$\text{or } \sim\overrightarrow{\langle\mathsf{zero}_{\ell}\mathsf{decOk}_{\ell+1}\ \beta\rangle}0_{\ell+1}c_m 0_{\ell+1}\sigma w\tau x \qquad (\text{rule for } \overrightarrow{\langle\mathsf{zero}^1_{\ell+1}\ \beta\rangle})$$

$$\Longleftrightarrow\quad \sim\overrightarrow{\langle\mathsf{zero}^1_{\ell+1}\ \beta\rangle}c_{m-1}0_{\ell+1}c_m 0_{\ell+1}\sigma w\tau x \qquad (\text{ind. hyp. on (d),(c) and } m > 1))$$

$$\Longleftrightarrow\quad \cdots$$

$$\Longleftrightarrow\quad \sim\overrightarrow{\langle\mathsf{dec}_{\ell}\mathsf{zero}^1_{\ell+1}\ \beta\rangle}0_{\ell+1}c_1\cdots 0_{\ell+1}c_m 0_{\ell+1}\sigma w\tau x$$

$$\text{or } \sim\overrightarrow{\langle\mathsf{zero}_{\ell}\mathsf{decOk}_{\ell+1}\ \beta\rangle}0_{\ell+1}c_1\cdots 0_{\ell+1}c_m 0_{\ell+1}\sigma w\tau x$$

$$\Longleftrightarrow\quad \sim\overrightarrow{\langle\mathsf{zero}^1_{\ell+1}\ \beta\rangle}c_0 0_{\ell+1}c_1\cdots 0_{\ell+1}c_m 0_{\ell+1}\sigma w\tau x$$

$$\text{or } \sim\overrightarrow{\langle\mathsf{decOk}_{\ell+1}\ \beta\rangle}c_0 0_{\ell+1}c_1\cdots 0_{\ell+1}c_m 0_{\ell+1}\sigma w\tau x \qquad (\text{ind. hyp. on (d),(c)})$$

$$\Longleftrightarrow\quad \sim\overrightarrow{\langle\mathsf{zero}^1_{\ell+1}\ \beta\rangle}c_0 0_{\ell+1}c_1\cdots 0_{\ell+1}c_m 0_{\ell+1}\sigma w\tau x$$

$$\text{or } (\sim\overrightarrow{\langle\beta\rangle}\underbrace{c_0 0_{\ell+1}c_1\cdots 0_{\ell+1}c_m 0_{\ell+1}}_{\text{zero }(l+1)\text{-counter}}\sigma w\tau x \text{ and } val(w) = 1 \quad (\text{part (b)})$$

Further we have

$$\sim\overrightarrow{\langle\mathsf{zero}^1_{\ell+1}\ \beta\rangle}c_0 0_{\ell+1}c_1\cdots 0_{\ell+1}c_m 0_{\ell+1}\sigma w\tau x$$

$$\Longleftrightarrow\quad \sim\overrightarrow{\langle\mathsf{dec}_{\ell}\ \mathsf{zero}^1_{\ell+1}\ \beta\rangle}0_{\ell+1}c_0 0_{\ell+1}c_1\cdots 0_{\ell+1}c_m 0_{\ell+1}\sigma w\tau x$$

$$\text{or } \sim\overrightarrow{\langle\mathsf{zero}_{\ell}\mathsf{decOk}_{\ell+1}\ \beta\rangle}0_{\ell+1}c_0 0_{\ell+1}c_1\cdots 0_{\ell+1}c_m 0_{\ell+1}\sigma w\tau x \qquad (\text{rule for } \overrightarrow{\langle\mathsf{zero}^1_{\ell+1}\ \beta\rangle})$$

$$\Longleftrightarrow\quad \textit{false} \qquad (\text{ind. hyp. on (d),(c)})$$

Combining this with the equivalences above yields part (c) for $\ell + 1$.

For part (d) we obtain the following equivalences:

$$
\begin{aligned}
\sim\overrightarrow{\langle\mathsf{dec}_{\ell+1}\beta\rangle}\sigma w\tau x &\iff \exists\sigma_m \in \Omega_{\ell+1} : \sim\overrightarrow{\langle\mathsf{ones}_\ell\mathsf{dec}^1_{\ell+1}\ \beta\rangle}\sigma_m\sigma w\tau x && \text{(rule for } \overrightarrow{\langle\mathsf{dec}_{\ell+1}\ \beta\rangle}) \\
&\iff \exists\sigma_m \in \Omega_{\ell+1} : \sim\overrightarrow{\langle\mathsf{dec}^1_{\ell+1}\ \beta\rangle}c_m\sigma_m\sigma w\tau x && \text{(ind. hyp. on (a))} \\
&\iff \exists\sigma_{m-1},\sigma_m \in \Omega_{\ell+1} : \\
&\qquad \sim\overrightarrow{\langle\mathsf{dec}_\ell\mathsf{dec}^1_{\ell+1}\ \beta\rangle}\sigma_{m-1}c_m\sigma_m\sigma w\tau x \\
&\qquad \text{or } \sim\overrightarrow{\langle\mathsf{zero}_\ell\mathsf{decOk}_{\ell+1}\ \beta\rangle}\sigma_{m-1}c_m\sigma_m\sigma w\tau x && \text{(rule for } \overrightarrow{\langle\mathsf{dec}^1_{\ell+1}\ \beta\rangle}) \\
&\iff \exists\sigma_{m-1},\sigma_m \in \Omega_{\ell+1} : \\
&\qquad \sim\overrightarrow{\langle\mathsf{dec}^1_{\ell+1}\ \beta\rangle}c_{m-1}\sigma_{m-1}c_m\sigma_m\sigma w\tau x && \text{(ind. hyp. on (d),(c),} m > 1) \\
&\iff \cdots \\
&\iff \exists\sigma_1,\ldots,\sigma_m \in \Omega_{\ell+1} : \\
&\qquad \sim\overrightarrow{\langle\mathsf{dec}^1_{\ell+1}\ \beta\rangle}c_1\sigma_1\cdots c_m\sigma_m\sigma w\tau x \\
&\iff \exists\sigma_0,\ldots,\sigma_m \in \Omega_{\ell+1} : \\
&\qquad \sim\overrightarrow{\langle\mathsf{dec}_\ell\mathsf{dec}^1_{\ell+1}\ \beta\rangle}\sigma_0 c_1\sigma_1\cdots c_m\sigma_m\sigma w\tau x \\
&\qquad \text{or } \sim\overrightarrow{\langle\mathsf{zero}_\ell\mathsf{decOk}_{\ell+1}\ \beta\rangle}\sigma_0 c_1\sigma_1\cdots c_m\sigma_m\sigma w\tau x && \text{(rule for } \overrightarrow{\langle\mathsf{dec}^1_{\ell+1}\rangle}) \\
&\iff \exists\sigma_0,\ldots,\sigma_m \in \Omega_{\ell+1} : \\
&\qquad \sim\overrightarrow{\langle\mathsf{dec}_\ell\mathsf{dec}^1_{\ell+1}\beta\rangle}\sigma_0 c_1\sigma_1\cdots c_m\sigma_m\sigma w\tau x \\
&\qquad \text{or } \sim\overrightarrow{\langle\mathsf{decOk}_{\ell+1}\ \beta\rangle}c_0\sigma_0 c_1\sigma_1\cdots c_m\sigma_m\sigma w\tau x && \text{(ind. hyp. on (c)} \\
&\iff \exists\sigma_0,\ldots,\sigma_m \in \Omega_{\ell+1} : \\
&\qquad \sim\overrightarrow{\langle\mathsf{dec}_\ell\mathsf{dec}^1_{\ell+1}\beta\rangle}\sigma_0 c_1\sigma_1\cdots c_m\sigma_m\sigma w\tau x \\
&\qquad \text{or } (\sim\overrightarrow{\langle\beta\rangle}c_0\sigma_0 c_1\sigma_1\cdots c_m\sigma_m\sigma w\tau x \\
&\qquad\qquad \text{and } val(w) = val(c_0\sigma_0\cdots c_m\sigma_m)+1) && (\star) \\
& && \text{(ind. hyp. on (b))}
\end{aligned}
$$

Further we have for all $\sigma_0,\ldots,\sigma_m \in \Omega_{\ell+1}$:

$$
\begin{aligned}
& \sim\overrightarrow{\langle\mathsf{dec}_\ell\mathsf{dec}^1_{\ell+1}\ \beta\rangle}\sigma_0 c_1\sigma_1\cdots c_m\sigma_m\sigma w\tau x \\
\iff\ & \sim\overrightarrow{\langle\mathsf{dec}^1_{\ell+1}\ \beta\rangle}c_0\sigma_0 c_1\sigma_1\cdots c_m\sigma_m\sigma w\tau x && \text{(ind. hyp. on (d))} \\
\iff\ & \exists\sigma_{-1} \in \Omega_{\ell+1} : \sim\overrightarrow{\langle\mathsf{dec}_\ell\mathsf{dec}^1_{\ell+1}\beta\rangle}\sigma_{-1}c_0\sigma_0 c_1\sigma_1\cdots c_m\sigma_m\sigma w\tau x \\
& \qquad \text{or } \sim\overrightarrow{\langle\mathsf{zero}_\ell\mathsf{decOk}_{\ell+1}\ \beta\rangle}\sigma_{-1}c_0\sigma_0 c_1\sigma_1\cdots c_m\sigma_m\sigma w\tau x \\
\iff\ & \textit{false} && \text{(ind. hyp. on (d),(c))}
\end{aligned}
$$

Hence $(\star)$ is equivalent to

$$
\exists\sigma_0,\ldots,\sigma_m \in \Omega_{\ell+1} : \sim\overrightarrow{\langle\beta\rangle}c_0\sigma_0 c_1\sigma_1\cdots c_m\sigma_m\sigma w\tau x \ \text{ and } \ val(w) = val(c_0\sigma_0\cdots c_m\sigma_m)+1
$$

which is in turn equivalent to

$$
val(w) \neq 0 \ \text{ and } \sim\overrightarrow{\langle\beta\rangle}v\sigma w\tau x, \text{ where } v \text{ is the } \ell\text{-counter with } val(v)+1 = val(w),
$$

which shows part (d) for $\ell+1$. $\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad\ \square$

4.3 Simulating a Transducer Machine

Returning to our overall reduction, let us recall we have fixed a deterministically terminating transducer machine $\mathcal{T} = (\mathsf{Tower}(k,n), T_1, T_2)$ with respect to which $0^{\mathsf{Tower}(k,n)}$ is a dead end, i.e. we have the following in total:

- $T_1, T_2 : \{0,1\}^* \to \Upsilon^*$ are letter-to-letter transducers,
- for each $z \in \{0,1\}^{\mathsf{Tower}(k,n)}$ there is at most one z' with $T_1(z) = T_2(z')$,
- assume $z_0, \ldots, z_t \in \{0,1\}^{\mathsf{Tower}(k,n)}$ such that $z_0 = 1^{\mathsf{Tower}(k,n)}$ and $T_1(z_i) = T_2(z_{i+1})$ for each
  $i \in [0, t-1]$, and z_t is a dead end (we defined $\mathsf{last}(\mathcal{T}) \overset{\text{def}}{=} z_t$), and
- $0^{\mathsf{Tower}(k,n)}$ is a dead end with respect to $\mathcal{T}$.

We include rules so that $\sim\overrightarrow{\langle\mathsf{start}\rangle}$ holds if and only if $\mathsf{last}(\mathcal{T}) = 0^{\mathsf{Tower}(k,n)}$, thus completing our reduction. The PDA $\mathcal{P}$ will be able to push $z_0, z_1, \ldots$ on the stack, where each word z_i is encoded as a k-counter, say d_i, in the obvious way: $z_i = \eta(d_i)$ where $\eta : \Omega^* \to \Omega_k^*$ denotes the homomorphism with $\eta(\sigma) = \sigma$ for $\sigma \in \Omega_k$ and $\eta(\sigma) = \varepsilon$ otherwise. We emphasize that in comparison to the counters that were present in the proof of Lemma 6 for each $i \in [0, t]$ we do not generally have $val(d_i) = i$: Instead we have $z_i = \eta(d_i)$, in particular the sequence $z_0, \ldots, z_t$ and thus the sequence $d_0, \ldots, d_t$ is determined. The d_i will be separated on the stack by the symbol $\$ \overset{\text{def}}{=} 0_{k+1}$. We include rules such that the following holds:

Lemma 7. *Let $x \in \Omega^*$ and let w_1, w_2, w_3 be k-counters.*

(a) Then $\sim\overrightarrow{\langle\mathsf{testFin}\rangle}w_1\$x$ iff $\eta(w_1) = 0^{\mathsf{Tower}(k,n)}$.
(b) Then $\sim\overrightarrow{\langle\mathsf{testTran}\rangle}w_3\$w_2\$w_1\$x$ iff $T_1(\eta(w_1)) = T_2(\eta(w_2))$.
(c) Then $\sim\overrightarrow{\langle\mathsf{start}\rangle}$ iff $z_t = 0^{\mathsf{Tower}(k,n)}$.

For Lemma 7 we include the following rules:

$$\overrightarrow{\langle\mathsf{start}\rangle} \circ\!\!\rightarrow \overrightarrow{\langle\mathsf{ones}_k\ \mathsf{fin}\rangle}\$ \qquad\qquad\text{push \$ and the encoded } z_0$$

$$\overrightarrow{\langle\mathsf{fin}\rangle} \overset{Def}{\circ\!\!\longrightarrow} \{\overrightarrow{\langle\mathsf{testFin}\rangle}, \overrightarrow{\langle\mathsf{next}\rangle}\$\} \qquad\qquad\text{test } 0^{\mathsf{Tower}(k,n)} \text{ OR do the next } z_i$$

$$\bullet\langle\mathsf{testFin}\rangle \left(\Omega_{\leq k-1}^*\Omega_k\right)^*\$ \xrightarrow{\{1_k\}\mapsto b\ \|\ (\Omega\setminus\{1_k\})\mapsto a} \bullet\langle\mathsf{popAll}\rangle \qquad\text{rule à la Lemma 3 (a) to test } 0^{\mathsf{Tower}(k,n)}$$

$$\langle\mathsf{testFin}\rangle_\bullet \left(\Omega_{\leq k-1}^*\Omega_k\right)^*\$ \xrightarrow{\Omega\mapsto a} \langle\mathsf{popAll}\rangle_\bullet \qquad\text{rule à la Lemma 3 (a) to test } 0^{\mathsf{Tower}(k,n)}$$

$$\bullet\langle\mathsf{popAll}\rangle\omega \overset{a}{\rightarrow} \bullet\langle\mathsf{popAll}\rangle \qquad\qquad\text{for all } \omega \in \Omega\text{: Erase stack content}$$

$$\langle\mathsf{popAll}\rangle_\bullet\omega \overset{a}{\rightarrow} \langle\mathsf{popAll}\rangle_\bullet \qquad\qquad\text{for all } \omega \in \Omega\text{: Erase stack content}$$

$$\overrightarrow{\langle\mathsf{next}\rangle} \overset{Def}{\circ\!\!\longrightarrow} \left\{\overrightarrow{\langle\mathsf{ones}_{k-1}\ \mathsf{next}^1\rangle}\sigma \mid \sigma \in \Omega_k\right\} \qquad\text{choose the next } z_i$$

$$\overrightarrow{\langle\mathsf{next}^1\rangle} \overset{Def}{\circ\!\!\longrightarrow} \left\{\overrightarrow{\langle\mathsf{dec}_{k-1}\ \mathsf{next}^1\rangle}\sigma,\right.$$

$$\left.\overrightarrow{\langle\mathsf{zero}_{k-1}\ \mathsf{tran}\rangle}\sigma \mid \sigma \in \Omega_k\right\}$$

$$\overrightarrow{\langle\mathsf{tran}\rangle} \overset{Att}{\circ\!\!\longrightarrow} \left\{\overrightarrow{\langle\mathsf{ones}_k\ \mathsf{testTran}\rangle}\$,\right. \qquad\qquad\text{test whether new } z_i \text{ is ok}$$

$$\left.\overrightarrow{\langle\mathsf{fin}\rangle}\right\} \qquad\qquad\text{OR continue}$$

$$\bullet\langle\mathsf{testTran}\rangle\Omega_{\leq k}^*\$\Omega_{\leq k}^*\$ \xrightarrow{\Omega\mapsto a} \bullet\langle\mathsf{testTran}^1\rangle \qquad\text{rule à la Lemma 3}$$

$$\langle\mathsf{testTran}\rangle_\bullet\Omega_{\leq k}^*\$ \xrightarrow{\Omega\mapsto aa} \langle\mathsf{testTran}^1\rangle_\bullet \qquad\text{rule à la Lemma 3}$$

$$\bullet\langle\mathsf{testTran}^1\rangle \left(\Omega_{\leq k-1}^*\Omega_k\right)^*\$ \xrightarrow{T_1\ \|\ ((\Omega\setminus\Omega_k)\mapsto a)} \bullet\langle\mathsf{stop}_k\rangle \qquad\text{rule à la Lemma 3 for } T_1$$

$$\langle\mathsf{testTran}^1\rangle_\bullet \left(\Omega_{\leq k-1}^*\Omega_k\right)^*\$ \xrightarrow{T_2\ \|\ ((\Omega\setminus\Omega_k)\mapsto a)} \langle\mathsf{stop}_k\rangle_\bullet \qquad\text{rule à la Lemma 3 for } T_2$$

Proof (of Lemma 7). Parts (a) and (b) are simple consequences of the above rules and Lemma 3 (a) and (b), respectively. For part (c) we have:

$$\sim\overrightarrow{\langle\mathsf{start}\rangle} \quad\Longleftrightarrow\quad \sim\overrightarrow{\langle\mathsf{fin}\rangle}d_0\$ \qquad\qquad \text{rule for } \overrightarrow{\langle\mathsf{start}\rangle} \text{ and Lemma 6 (a)}$$

$$\Longleftrightarrow\quad \eta(d_0) = 0^{\mathsf{Tower}(k,n)} \text{ or } \sim\overrightarrow{\langle\mathsf{next}\rangle}\$d_0\$ \qquad \text{rule for } \overrightarrow{\langle\mathsf{fin}\rangle} \text{ and part (a)}$$

By the rules for $\overrightarrow{\langle\mathsf{next}\rangle}$ and $\overrightarrow{\langle\mathsf{next}^1\rangle}$ and reasoning as in Lemma 6 we have

$$\sim\overrightarrow{\langle\mathsf{next}\rangle}\$d_0\$ \quad\Longleftrightarrow\quad \text{there is a } k\text{-counter } w_1 \text{ with } \sim\overrightarrow{\langle\mathsf{tran}\rangle}w_1\$d_0\$.$$

By the rules for $\overrightarrow{\langle\mathsf{tran}\rangle}$ and part (b) we have

$$\sim\overrightarrow{\langle\mathsf{tran}\rangle}w_1\$d_0\$ \quad\Longleftrightarrow\quad w_1 = d_1 \text{ and } \sim\overrightarrow{\langle\mathsf{fin}\rangle}w_1\$d_0\$.$$

It follows that:

$$\sim\overrightarrow{\langle\mathsf{next}\rangle}\$d_0\$ \quad\Longleftrightarrow\quad \sim\overrightarrow{\langle\mathsf{fin}\rangle}d_1\$d_0\$$$

$$\Longleftrightarrow\quad \eta(d_1) = 0^{\mathsf{Tower}(k,n)} \text{ or } \sim\overrightarrow{\langle\mathsf{next}\rangle}\$d_1\$d_0\$ \qquad \text{rule for } \overrightarrow{\langle\mathsf{fin}\rangle} \text{ and part (a)}$$

Combining this with the equivalences above we obtain:

$$\sim\overrightarrow{\langle\mathsf{start}\rangle} \quad\Longleftrightarrow\quad \bigvee_{i=0}^{1} \eta(d_i) = 0^{\mathsf{Tower}(k,n)} \text{ or } \sim\overrightarrow{\langle\mathsf{next}\rangle}\$d_1\$d_0\$$$

By iterating this reasoning, we obtain:

$$\sim\overrightarrow{\langle\mathsf{start}\rangle} \quad\Longleftrightarrow\quad \bigvee_{i=0}^{t} \eta(d_i) = 0^{\mathsf{Tower}(k,n)} \text{ or } \sim\overrightarrow{\langle\mathsf{next}\rangle}\$d_t\$d_{t-1}\$\cdots d_0\$$$

$$\Longleftrightarrow\quad \bigvee_{i=0}^{t} z_i = 0^{\mathsf{Tower}(k,n)} \text{ or } \sim\overrightarrow{\langle\mathsf{next}\rangle}\$d_t\$d_{t-1}\$\cdots d_0\$$$

$$\Longleftrightarrow\quad z_t = 0^{\mathsf{Tower}(k,n)} \text{ or } \sim\overrightarrow{\langle\mathsf{next}\rangle}\$d_t\$d_{t-1}\$\cdots d_0\$,$$

where the last equivalence follows from the fact that $0^{\mathsf{Tower}(k,n)}$ is a dead end. By reasoning as above we have:

$$\sim\overrightarrow{\langle\mathsf{next}\rangle}\$d_t\$\cdots d_0\$ \quad\Longrightarrow\quad \text{there is a } k\text{-counter } w \text{ with } \sim\overrightarrow{\langle\mathsf{tran}\rangle}w\$d_t\$\cdots d_0\$$$

$$\Longrightarrow\quad T_1(\eta(d_t)) = T_2(\eta(w)) ,$$

which is false, by definition of t. We conclude that $\sim\overrightarrow{\langle\mathsf{start}\rangle}$ holds iff $z_t = 0^{\mathsf{Tower}(k,n)}$. $\qquad\square$

By Lemma 7 (c) we have completed the reduction, and hence the proof of Theorem 1.

4.4 Normedness

We can strengthen Theorem 1. Given a PDA $\mathcal{P}$ with control state set Q, we say a state s of $\mathcal{S}(\mathcal{P})$ is *normed* if every state t that is reachable from s can reach some deadlock t', where $t' = q$ for some $q \in Q$ (i.e. the stack is empty).

Theorem 8. *PDA bisimilarity for is nonelementary, even when the initial states are normed.*

Proof. Recall that (5) requires for $x \in \Omega^*$ and $\sigma \in \Omega_{\ell+1}$ and ℓ-counters w that we have

$$\bullet\langle\mathsf{stop}_\ell\rangle x \sim \langle\mathsf{stop}_\ell\rangle_\bullet w\sigma x \;.$$

We satisfied this by not giving any rules for $_\bullet\langle\mathsf{stop}_\ell\rangle$ and $\langle\mathsf{stop}_\ell\rangle_\bullet$, thus creating deadlocks. Thus we allowed to reach states that are not normed. We show that the construction can be amended to avoid the latter. Assume that $x = w'\sigma'x'$ holds for an ℓ-counter w', $\sigma' \in \Omega_{\ell+1}$ and $x' \in \Omega^*$. Add the following rules:

$$_\bullet\langle\mathsf{stop}_\ell\rangle\Omega^*_{\leq\ell}\Omega_{\ell+1} \xrightarrow{\Omega\mapsto aa} {}_\bullet\langle\mathsf{popAll}\rangle$$

$$\langle\mathsf{stop}_\ell\rangle_\bullet\Omega^*_{\leq\ell}\Omega_{\ell+1}\Omega^*_{\leq\ell}\Omega_{\ell+1} \xrightarrow{\Omega\mapsto a} \langle\mathsf{popAll}\rangle_\bullet$$

and recall that rules for $_\bullet\langle\mathsf{popAll}\rangle$ and $\langle\mathsf{popAll}\rangle_\bullet$ were given previously. Intuitively, $w'\sigma'$ is popped off the left stack with "half" speed, while $w\sigma w'\sigma'$ is popped off the right stack with "full" speed. Afterwards the stacks are of equal height and can be fully erased. Now (5) is satisfied, and these states are normed. For the latter fact, observe all reachable stacks are of the form $((\ldots((\Omega_0^*\Omega_1)^*\Omega_2)^*\ldots)^*\$)^*$ and $_\bullet\langle\mathsf{popAll}\rangle$ or $\langle\mathsf{popAll}\rangle_\bullet$ is always reachable. The only remaining problem is to guarantee that, whenever we require $_\bullet\langle\mathsf{stop}_\ell\rangle x \sim \langle\mathsf{stop}_\ell\rangle_\bullet w\sigma x$, x is indeed prefixed by $w'\sigma'$. Note that stop_k is introduced in the rule for $\mathsf{testTran}$ and other occurrences of stop_ℓ are used in the counter-related rules based on $\mathsf{testDec}_\ell$. Consequently, if the above requirement is not already satisfied, w must be the ones ℓ-counter. To prevent this from happening, for $\mathsf{testTran}$ we shall add an extra ones k-counter at the beginning of the simulations. For $\mathsf{testDec}$, we shall eliminate the need for decrement tests involving ones counters by introducing a new symbol zOnes that will correspond to the predecessor of a ones counter.

This can be achieved by the following modifications:

– Replace $\overrightarrow{\langle\mathsf{start}\rangle} \multimap \overrightarrow{\langle\mathsf{ones}_k\ \mathsf{fin}\rangle}\$$ with $\overrightarrow{\langle\mathsf{start}\rangle} \multimap \overrightarrow{\langle\mathsf{ones}_k\ \mathsf{start}^0\rangle}\$$ and $\overrightarrow{\langle\mathsf{start}^0\rangle} \multimap \overrightarrow{\langle\mathsf{ones}_k\ \mathsf{fin}\rangle}\$$;
  i.e., push one additional ones k-counter and $\$$ in the beginning.
– Extend B by symbols zOnes_ℓ with $\ell \in [0, k+2]$, whose role is to push an ℓ-counter w with $val(w) = \mathsf{Tower}(\ell+1, n) - 2$. (This value is one less than the value of a ones ℓ-counter.)
– Replace $\overrightarrow{\langle\mathsf{ones}_\ell\ \beta\rangle} \multimap \overrightarrow{\langle\mathsf{ones}_{\ell-1}\ \mathsf{ones}_\ell^1\ \beta\rangle}1_\ell$ with

$$\overrightarrow{\langle\mathsf{ones}_\ell\ \beta\rangle} \multimap \overrightarrow{\langle\mathsf{ones}_{\ell-1}\ \mathsf{ones}_\ell^0\ \beta\rangle}1_\ell \quad\text{and add}$$

$$\overrightarrow{\langle\mathsf{ones}_\ell^0\ \beta\rangle} \multimap \overrightarrow{\langle\mathsf{zOnes}_{\ell-1}\ \mathsf{ones}_\ell^1\ \beta\rangle}1_\ell\;.$$

Similarly, replace $\overrightarrow{\langle\mathsf{dec}_\ell\ \beta\rangle} \xrightarrow{Def} \left\{\overrightarrow{\langle\mathsf{ones}_{\ell-1}\ \mathsf{dec}_\ell^1\ \beta\rangle}\sigma \mid \sigma \in \Omega_\ell\right\}$ with

$$\overrightarrow{\langle\mathsf{dec}_\ell\ \beta\rangle} \xrightarrow{Def} \left\{\overrightarrow{\langle\mathsf{ones}_{\ell-1}\ \mathsf{dec}_\ell^0\ \beta\rangle}\sigma \mid \sigma \in \Omega_\ell\right\} \quad\text{and add}$$

$$\overrightarrow{\langle\mathsf{dec}_\ell^0\ \beta\rangle} \xrightarrow{Def} \left\{\overrightarrow{\langle\mathsf{zOnes}_{\ell-1}\ \mathsf{dec}_\ell^1\ \beta\rangle}\sigma \mid \sigma \in \Omega_\ell\right\}\;.$$

Similarly, replace $\overrightarrow{\langle\mathsf{zero}_\ell\ \beta\rangle} \multimap \overrightarrow{\langle\mathsf{ones}_{\ell-1}\ \mathsf{zero}_\ell^1\ \beta\rangle}0_\ell$ with

$$\overrightarrow{\langle\mathsf{zero}_\ell\ \beta\rangle} \multimap \overrightarrow{\langle\mathsf{ones}_{\ell-1}\ \mathsf{zero}_\ell^0\ \beta\rangle}0_\ell \quad\text{and add}$$

$$\overrightarrow{\langle\mathsf{zero}_\ell^0\ \beta\rangle} \multimap \overrightarrow{\langle\mathsf{zOnes}_{\ell-1}\ \mathsf{zero}_\ell^1\ \beta\rangle}0_\ell\;.$$

– For $\ell \geq 1$ add the following rules:

$$\overrightarrow{\langle\mathsf{zOnes}_\ell\ \beta\rangle} \multimap \overrightarrow{\langle\mathsf{ones}_{\ell-1}\ \mathsf{zOnes}_\ell^0\ \beta\rangle}1_\ell$$

$$\overrightarrow{\langle\mathsf{zOnes}_\ell^0\ \beta\rangle} \multimap \overrightarrow{\langle\mathsf{zOnes}_{\ell-1}\ \mathsf{zOnes}_\ell^1\ \beta\rangle}1_\ell$$

$$\overrightarrow{\langle\mathsf{zOnes}_\ell^1\ \beta\rangle} \xrightarrow{Def} \left\{\overrightarrow{\langle\mathsf{dec}_{\ell-1}\ \mathsf{zOnes}_\ell^1\ \beta\rangle}1_\ell, \overrightarrow{\langle\mathsf{zero}_{\ell-1}\ \beta\rangle}0_\ell\right\}$$

Finally, add $\overrightarrow{\langle\mathsf{zOnes}_0\ \beta\rangle} \multimap \overrightarrow{\langle\beta\rangle}0_0 1_0^{n-1}$.

16

By these modifications, whenever we require $\bullet\langle\mathsf{stop}_\ell\rangle x \sim \langle\mathsf{stop}_\ell\rangle\bullet w\sigma x$ in our proofs, the word x will start with an ℓ-counter and an $\Omega_{\ell+1}$-symbol. $\qquad\square$

References

1. Stanislav Böhm, Stefan Göller, and Petr Jančar. Bisimilarity of one-counter processes is PSPACE-complete. In *Proc. of CONCUR*, volume 6269 of *Lecture Notes in Computer Science*, pages 177–191. Springer, 2010.
2. Christopher Broadbent and Stefan Göller. On Bisimilarity on Higher-Order Pushdown Automata: Undecidability at Order Two. In *FSTTCS*, 2012.
3. O. Burkart, D. Caucal, and B. Steffen. An elementary bisimulation decision procedure for arbitrary context-free processes. In *Mathematical Foundations of Computer Science*, volume 969 of *LNCS*, pages 423–433, 1995.
4. Petr Jancar. Decidability of DPDA language equivalence via first-order grammars. In *LICS*, pages 415–424. IEEE, 2012.
5. David Janin and Igor Walukiewicz. On the Expressive Completeness of the Propositional mu-Calculus with Respect to Monadic Second Order Logic. In *Proc. of CONCUR*, volume 1119 of *Lecture Notes in Computer Science*, pages 263–277. Springer, 1996.
6. Petr Jančar. Strong bisimilarity on basic parallel processes is pspace-complete. In *Proc. of LICS*, pages 218–227. IEEE Computer Society, 2003.
7. Petr Jančar and Jirí Srba. Undecidability of bisimilarity by defender's forcing. *J. ACM*, 55(1), 2008.
8. Stefan Kiefer. BPA Bisimilarity is EXPTIME-hard. *CoRR*, abs/1205.7041, 2012.
9. A. Kucera and R. Mayr. On the complexity of checking semantic equivalences between pushdown processes and finite-state processes. *Information and Computation*, 208(7):772–796, 2010.
10. A. Kučera and P. Jančar. Equivalence-checking on infinite-state systems: Techniques and results. *Theory and Practice of Logic Programming*, 6(3):227–264, 2006.
11. R. Milner. *Communication and Concurrency*. Prentice Hall, 1989.
12. C. H. Papadimitriou. *Computational Complexity*. Addison-Wesley, 1994.
13. G. Sénizergues. L(A)=L(B)? Decidability results from complete formal systems. *Theoretical Computer Science*, 251(1-2):1–166, 2001.
14. G. Sénizergues. The bisimulation problem for equational graphs of finite out-degree. *SIAM Journal on Computing*, 34(5):1025–1106, 2005.
15. Colin Stirling. Deciding DPDA Equivalence Is Primitive Recursive. In *Proc. of ICALP*, volume 2380 of *Lecture Notes in Computer Science*, pages 821–832. Springer, 2002.
16. Larry J. Stockmeyer. *The complexity of decision problems in automata and logic*. PhD thesis, Massachusetts Institute of Technology, Cambridge, MA, 1974.
17. Johan van Benthem. *Modal Correspondence Theory*. PhD thesis, University of Amsterdam, 1976.
18. Rob J. van Glabbeek. The linear time-branching time spectrum (extended abstract). In *CONCUR*, volume 458 of *Lecture Notes in Computer Science*, pages 278–297. Springer, 1990.