Language Equivalence of Probabilistic Pushdown Automata

Vojtěch Forejt[a], Petr Jančar[b], Stefan Kiefer[a], James Worrell[a]

[a] *Department of Computer Science, University of Oxford, UK*
[b] *Dept of Computer Science, FEI, Techn. Univ. Ostrava, Czech Republic*

Abstract

We study the language equivalence problem for probabilistic pushdown automata (pPDA) and their subclasses. We show that the problem is interreducible with the multiplicity equivalence problem for context-free grammars, the decidability of which has been open for several decades. Interreducibility also holds for pPDA with one control state.
In contrast, for the case of a one-letter input alphabet we show that pPDA language equivalence (and hence multiplicity equivalence of context-free grammars) is in PSPACE and at least as hard as the polynomial identity testing problem.

Keywords: pushdown systems, language equivalence, probabilistic systems

1. Introduction

Equivalence checking is the problem of determining whether two systems are semantically identical. This is an important question in automated verification and, more generally, represents a line of research that can be traced back to the inception of theoretical computer science. A great deal of work in this area has been devoted to the complexity of *language equivalence* for various classes of infinite-state systems based on grammars and automata, such as basic process algebras (BPA) and pushdown processes. We mention in particular the landmark result showing the decidability of language equivalence for deterministic pushdown automata (dPDA) [24]; the problem is well-known to be undecidable for general (nondeterministic) PDA.

An input word determines a unique computation of a dPDA, whereas the computation of a PDA on an input word can have many branches. In this paper we are concerned with *probabilistic pushdown automata* (pPDA), where we only allow probabilistic branching. Here two pPDA are language equivalent if they accept each word with the same probability. The decidability of the language equivalence problem for pPDA is still open, even in the case with no ε-transitions, to which we restrict ourselves in this paper.

The language theory of probabilistic pushdown automata has been studied in [1], where their equivalence with stochastic context-free grammars (CFGs)

is proved. There is also a growing body of work concerning the complexity of model checking and equivalence checking of probabilistic pushdown automata, probabilistic one-counter machines and probabilistic BPA (see, e.g., [5, 10, 11, 13]).

It was shown recently in [17] that the language equivalence problem for probabilistic visibly pushdown automata is logspace equivalent to the problem of *polynomial identity testing*, that is, determining whether a polynomial presented as an arithmetic circuit is identically zero. The latter problem is known to be in coRP.

The contribution of this paper is the following. For general pPDA we show that language equivalence is polynomially interreducible with *multiplicity equivalence* of CFGs. The latter problem asks whether in two given grammars every word has the same number of derivation trees. The decidability question for multiplicity equivalence is a long-standing open problem in theory of formal languages [21, 19, 18, 15]. Our construction works by turning nondeterministic branches of a CFG into carefully designed probabilistic transitions of a pPDA, and vice versa. A consequence of this reduction is that the equivalence problem for pPDA is polynomially reducible to the equivalence problem for pPDA with one control state. We note that a corresponding polynomial reduction from the general case to the one-state case would be a breakthrough in the case of deterministic PDA since one-state dPDA equivalence is known to be in P (see [14], or [8] for the best known upper bound).

We further show that in the case of a one-letter input alphabet the language equivalence problem is decidable in polynomial space. We use the fact that in this case the problem reduces to comparing distributions of *termination probabilities* within i steps ($i = 0, 1, 2, \dots$). By using an equation system for generating functions we reduce the latter problem to the decision problem for the existential fragment of the theory of the reals (which is known to be in PSPACE but not known to be PSPACE-complete). Moreover, we show that the hardness result from [17] carries over; i.e., language equivalence for one-letter pPDA is at least as hard as the polynomial identity testing. Very recent work [7] considers (non)probabilistic dPDA with a one-letter input alphabet, allowing for ε-transitions. They show, among other results, that the equivalence problem for such dPDA is P-complete.

As a byproduct of the mentioned results, we obtain that multiplicity equivalence of CFG with one-letter input alphabet is in PSPACE. The previously known decidability result, which is based on *elimination theory* for systems of polynomial equations, did not provide any complexity bound, see [19, 18, 15] and the references therein.

2. Definitions and results

By $\mathbb{N}, \mathbb{Q}, \mathbb{R}$ we denote the set of nonnegative integers, the set of rationals, and the set of reals, respectively. We denote the set of *words* over a finite alphabet Σ by Σ^*. We denote the *empty word* by ε and write $\Sigma^+ = \Sigma^* \setminus \{\varepsilon\}$.

By $|w|$ we denote the length of $w \in \Sigma^*$, so that $|\varepsilon| = 0$. For $k \in \mathbb{N}$ we put $\Sigma^{\leq k} = \{w \in \Sigma^*; |w| \leq k\}$.

Given a finite or countable set A, a *probability distribution* on A is a function $d : A \to [0,1] \cap \mathbb{Q}$ such that $\sum_{a \in A} d(a) = 1$. The *support* of a probability distribution d is the set $\mathit{support}(d) := \{a \in A : d(a) > 0\}$. The set of all probability distributions on A is denoted by $\mathcal{D}(A)$. A *Dirac distribution* is one whose support is a singleton.

2.1. Probabilistic Labelled Transition Systems

A *probabilistic labelled transition system (pLTS)* is a tuple $\mathcal{S} = (S, \Sigma, \to)$, where S is a finite or countable set of *states*, Σ is a finite *input alphabet*, whose elements are also called *actions*, and $\to \subseteq S \times \Sigma \times \mathcal{D}(S)$ is a *transition relation* satisfying that for each pair (s, a) there is at most one d such that $(s, a, d) \in \to$. We write $s \xrightarrow{a} d$ to say that $(s, a, d) \in \to$, and $s \xrightarrow{a,x} s'$ when there is $s \xrightarrow{a} d$ such that $d(s') = x$. We also write $s \to s'$ to say that there exists a transition $s \xrightarrow{a} d$ with $s' \in \mathit{support}(d)$. We say that an *action* a is *enabled in a state* $s \in S$ if $s \xrightarrow{a} d$ for some d; otherwise a is *disabled in* s. A *state* $s \in S$ is *terminating* if no action is enabled in s.

Let $\mathcal{S} = (S, \Sigma, \to)$ be a pLTS. An *execution* on a word $a_1 a_2 \dots a_k \in \Sigma^*$, starting in a given state s_0, is a finite sequence $s_0 \xrightarrow{a_1,x_1} s_1 \xrightarrow{a_2,x_2} s_2 \cdots \xrightarrow{a_k,x_k} s_k$. Given s_0 and $a_1 a_2 \dots a_k$, the probability of such an execution is $\prod_{i=1}^{k} x_i$.

2.2. Probabilistic Pushdown Automata

A *probabilistic pushdown automaton* (pPDA) is a tuple $\Delta = (Q, \Gamma, \Sigma, \hookrightarrow)$ where Q is a finite set of *control states*, Γ is a finite *stack alphabet*, Σ is a finite *input alphabet*, and $\hookrightarrow \subseteq Q \times \Gamma \times \Sigma \times \mathcal{D}(Q \times \Gamma^{\leq 2})$ is a finite set of *rules*. We require that for each $(q, X, a) \in Q \times \Gamma \times \Sigma$ there be at most one distribution d such that $(q, X, a, d) \in \hookrightarrow$. We write $qX \overset{a}{\hookrightarrow} d$ to denote $(q, X, a, d) \in \hookrightarrow$; informally speaking, in the control state q with X at the top of the stack we can perform an a-transition to the distribution d.

A *configuration* of a pPDA $\Delta = (Q, \Gamma, \Sigma, \hookrightarrow)$ is a pair $(q, \beta) \in Q \times \Gamma^*$; we often write $q\beta$ instead of (q, β). We write $qX \overset{a,x}{\hookrightarrow} r\beta$ if $qX \overset{a}{\hookrightarrow} d$ where $d(r\beta) = x$.

When speaking of the *size* of Δ, we assume that the probabilities in the transition relation are given as quotients of integers written in binary.

A pPDA $\Delta = (Q, \Gamma, \Sigma, \hookrightarrow)$ generates a pLTS $\mathcal{S}(\Delta) = (Q \times \Gamma^*, \Sigma, \to)$ as follows. For each $\beta \in \Gamma^*$, a rule $qX \overset{a}{\hookrightarrow} d$ of Δ induces a transition $qX\beta \xrightarrow{a} d'$ in $\mathcal{S}(\Delta)$, where $d' \in \mathcal{D}(Q \times \Gamma^*)$ is defined by $d'(p\alpha\beta) = d(p\alpha)$ for all $p \in Q$ and $\alpha \in \Gamma^{\leq 2}$ (and thus d' is 0 elsewhere). We note that all configurations with the empty stack, written as $p\varepsilon$ or just as p, are terminating states in $\mathcal{S}(\Delta)$. (Later we will assume that the empty-stack configurations are the only terminating states.)

The probability that Δ accepts a word $w \in \Sigma^*$ from a configuration $q\alpha$ is the sum of the probabilities of all executions on w, starting in $q\alpha$ in $\mathcal{S}(\Delta)$, that end in a configuration with the empty stack. We denote this probability by $\mathcal{P}^{\Delta}_{q\alpha}(w)$.

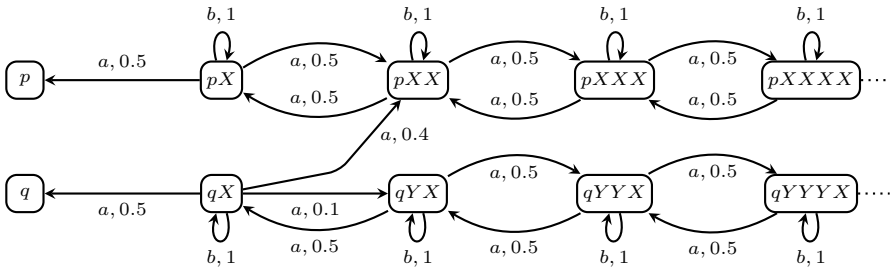

Figure 1: A fragment of $\mathcal{S}(\Delta)$ from Example 1.

A *probabilistic basic process algebra (pBPA)* Δ is a pPDA with only one control state. In this case we often write just α instead of $q\alpha$ for a configuration.

2.3. The Language Equivalence Problem

We study the *language equivalence problem* for pPDA. The problem asks whether two configurations $q_1\alpha_1$ and $q_2\alpha_2$ of a given pPDA Δ accept each input word with the same probability, i.e., whether the functions $\mathcal{P}^{\Delta}_{q_1\alpha_1}(\cdot)$ and $\mathcal{P}^{\Delta}_{q_2\alpha_2}(\cdot)$ are the same. If yes, we also say that $q_1\alpha_1$ and $q_2\alpha_2$ are *equivalent* in Δ.

Example 1. *Consider the pPDA* $\Delta = (\{p, q\}, \{X, Y\}, \{a, b\}, \hookrightarrow)$ *with the following rules:*

$$pX \xhookrightarrow{a,0.5} pXX \qquad pY \xhookrightarrow{a,1} pY \qquad qX \xhookrightarrow{a,0.1} qYX \qquad qY \xhookrightarrow{a,0.5} qYY$$
$$pX \xhookrightarrow{a,0.5} p\varepsilon, \qquad pY \xhookrightarrow{b,1} p\varepsilon \qquad qX \xhookrightarrow{a,0.5} q\varepsilon \qquad qY \xhookrightarrow{a,0.5} q\varepsilon$$
$$pX \xhookrightarrow{b,1} pX \qquad qX \xhookrightarrow{a,0.4} pXX \qquad qX \xhookrightarrow{b,1} qX \qquad qY \xhookrightarrow{b,1} qY$$

The restriction of Δ *to the control state* p *yields a pBPA. A fragment of the pLTS* $\mathcal{S}(\Delta)$ *is shown in Figure 1. The configurations* pXX *and* qYX *are equivalent in* Δ*, since for every word* w *we have* $\mathcal{P}^{\Delta}_{pXX}(w) = \mathcal{P}^{\Delta}_{qYX}(w)$*; this can be derived by observing that for all words* w *and* $i \geq 1$ *the probability of being in* pX^i *or* $qY^{i-1}X$ *after reading* w *is the same independently of whether we start from* pXX *or from* qYX *(a formal proof of this observation can be given by a straightforward induction on the length of* w*).*

In what follows we impose two restrictions on the language equivalence problem; both are without loss of generality. We say that a pPDA $\Delta = (Q, \Gamma, \Sigma, \hookrightarrow)$ is *non-blocking* if for each $(q, X, a) \in Q \times \Gamma \times \Sigma$ there is (precisely) one distribution d such that $qX \xhookrightarrow{a} d$; hence each action is disabled only in the empty-stack configurations in $\mathcal{S}(\Delta)$. We assume that our pPDA are non-blocking. Given an arbitrary pPDA $\Delta = (Q, \Gamma, \Sigma, \hookrightarrow)$ we obtain an equivalent non-blocking pPDA

Δ' as follows: we add a fresh stack symbol $\perp$ and for every (q, X, a) where there is no d such that $(q, X, a, d) \in \hookrightarrow$ (which includes the case $X = \perp$) we add the rule (q, X, a, d) where $d(q\perp) = 1$. Hence $\mathcal{P}^{\Delta'}_{q\alpha}(w) = 0$ if α contains $\perp$, and $\mathcal{P}^{\Delta}_{q\alpha}(w) = \mathcal{P}^{\Delta'}_{q\alpha}(w)$ for all $q \in Q$, $\alpha \in \Gamma^*$, $w \in \Sigma^*$.

We further suppose that the initial configurations $q_1\alpha_1$ and $q_2\alpha_2$ satisfy that $\alpha_1 = X_1$ and $\alpha_2 = X_2$ for some $X_1, X_2 \in \Gamma$. The general instance with $q_1\alpha_1$ and $q_2\alpha_2$ can be reduced to this form by adding some auxiliary stack symbols and rules, whose number is proportional to $k = \max\{|\alpha_1|, |\alpha_2|\}$: we just arrange that some freshly added configurations q_1Y_1, q_2Y_2 have the only possibility to move to $q_1\alpha_1$, $q_2\alpha_2$, respectively, by a fixed word a^k.

2.4. Grammars and the multiplicity problem

A *context-free grammar*, a *grammar* for short, is a tuple $G = (V, \Sigma, R, S)$ where V is a finite set of *nonterminals* (or *variables*), Σ is a finite set of *terminals*, $R \subseteq V \times (V \cup \Sigma)^+$ is a finite set of *production rules*, and $S \in V$ is a *start symbol*. We write production rules in the form $A \to \alpha$ where $\alpha \neq \varepsilon$, i.e., we assume that grammars are *ε-free*. The relation $\Rightarrow$ on $(V \cup \Sigma)^*$, capturing a *derivation step*, is defined as follows: if $A \to \alpha$ is in R then $\beta A\gamma \Rightarrow \beta\alpha\gamma$ (for any $\beta, \gamma \in (V \cup \Sigma)^*$). A sequence $\alpha_0 \Rightarrow \alpha_1 \Rightarrow \cdots \Rightarrow \alpha_k$ is a *derivation*, of α_k from α_0. A derivation step $\beta A\gamma \Rightarrow \beta\alpha\gamma$ is a *leftmost derivation step* if $\beta \in \Sigma^*$. A *derivation* $\alpha_0 \Rightarrow \alpha_1 \Rightarrow \cdots \Rightarrow \alpha_k$ is *leftmost* if $\alpha_i \Rightarrow \alpha_{i+1}$ is a leftmost step for each $i \in \{0, 1, \ldots, k-1\}$.

For a grammar $G = (V, \Sigma, R, S)$ and a word $w \in \Sigma^*$ we define the *multiplicity* $m_G(w) \in \mathbb{N} \cup \{\infty\}$ of w in G as the number of distinct leftmost derivations of w from S. We say that G is a *finite-multiplicity grammar* if $m_G(w) < \infty$ for all $w \in \Sigma^*$. Two finite-multiplicity grammars G_1, G_2 are said to be *multiplicity equivalent* if $m_{G_1}(w) = m_{G_2}(w)$ for all $w \in \Sigma^*$.

If a grammar has a rule $S \to S$ then $m_G(w) \in \{0, \infty\}$ for all $w \in \Sigma^*$, and multiplicity equivalence coincides with classical equivalence of grammars. Therefore multiplicity equivalence is undecidable in general. However for finite-multiplicity grammars, the decidability of multiplicity equivalence is a long-standing open problem [19]. This equivalence is known to be decidable only for subclasses of grammars, e.g., for *unary* grammars, i.e., grammars with $|\Sigma| = 1$, see [19, 18, 15] and the references therein.

2.5. Results

The results of this paper are captured by the next two theorems.

Theorem 1. *The following problems are interreducible in polynomial time, without changing the input alphabet or terminal alphabet, respectively:*

1. *language equivalence for pPDA;*
2. *language equivalence for pBPA;*
3. *multiplicity equivalence for (ε-free) finite-multiplicity grammars.*

Theorem 2. *The language equivalence problem for pPDA with a one-letter input alphabet, the language equivalence problem for pBPA with a one-letter input alphabet, and the multiplicity problem for (ε-free) finite-multiplicity grammars with a one-letter terminal alphabet, are all:*

1. *in PSPACE, and*
2. *at least as hard as polynomial identity testing (the ACIT problem, see the next subsection).*

2.6. Arithmetic Circuit Identity Testing

Recall that an *arithmetic circuit* is a finite directed acyclic multigraph C whose vertices, called *gates*, have indegree 0 or 2. Vertices of indegree 0 are called *input gates*; each input gate is labelled with 0, 1, or a variable from the set $\{x_i : i \in \mathbb{N}\}$. Vertices of indegree 2 are called *internal gates*; each such gate is labelled with one of the arithmetic operations $+$, $*$ or $-$. Since C is a multigraph, both inputs of a given gate can stem from the same source. We assume that there is a unique gate with outdegree 0 called the *output*. Any circuit C has a naturally related polynomial pol_C. A circuit C is *variable-free* if all inputs gates are labelled 0 or 1; pol_C is constant in this case.

The *Arithmetic Circuit Identity Testing (ACIT)* problem asks if pol_C is the zero polynomial, for a given circuit C. ACIT is known to be in coRP but it is open if it is in P; it is even open if there is a sub-exponential algorithm for this problem [2]. Utilising the fact that a variable-free arithmetic circuit of size $O(n)$ can compute 2^{2^n}, Allender *et al.* [2] give a logspace reduction of the general ACIT problem to the special case of variable-free circuits. Furthermore, ACIT can be reformulated as the problem of deciding whether two variable-free circuits using only the arithmetic operations $+$ and $*$ compute the same number [2].

2.7. Overview of the next sections

Theorem 1 is shown by adapting several existing constructions, and is postponed to Section 4. We start with proving Theorem 2 in Section 3: in Section 3.1 we show membership in PSPACE, and then show a polynomial reduction yielding the hardness result in Section 3.2.

3. Language Equivalence of pPDA with one input letter

In this section we consider *unary pPDA* $\Delta = (Q, \Gamma, \{a\}, \hookrightarrow)$, i.e., those whose input alphabet is a singleton $\{a\}$. In unary pPDA the next configuration is probabilistically determined solely by the current configuration. Here we elide the letter a in transitions, writing $qX \overset{p}{\hookrightarrow} r\alpha$ instead of $qX \overset{a,p}{\hookrightarrow} r\alpha$, and $q\beta \overset{p}{\to} r\gamma$ instead of $q\beta \overset{a,p}{\longrightarrow} r\gamma$.

3.1. Membership in PSPACE

In this subsection we prove the following lemma, establishing Point 1. in Theorem 2:

Lemma 1. *The language equivalence problem for unary pPDA is in PSPACE.*

We show this by a polynomial reduction to the decision problem for $ExTh(\mathbb{R})$, the existential fragment of the first-order theory of the reals, which is in PSPACE but not known to be PSPACE-hard. Hence language equivalence for unary pPDA is not PSPACE-hard unless $ExTh(\mathbb{R})$ is PSPACE-complete.

We will first note that in the unary case language equivalence coincides with the equality of termination-time distributions. For comparing two distributions, i.e., two countable sequences of nonnegative real numbers from $[0, 1]$, it is convenient to use the framework of generating functions. This allows us to create a system of equations with precisely one solution in the case of almost surely terminating unary pBPA. The equivalence question will thus reduce to deciding truth in $ExTh(\mathbb{R})$. The case of general unary pBPA is then handled by a reduction to the case of almost surely terminating pBPA. The result can then be extended to unary pPDA by using the direction 2.⇒1. of Theorem 1 (to be proven in Section 4).

3.1.1. Distribution of termination time of runs

Let us assume a fixed unary (non-blocking) pPDA $\Delta = (Q, \Gamma, \{a\}, \hookrightarrow)$. We note that our model of unary pPDA is essentially equivalent to a model which is also called "pPDA" in the literature, see e.g., [4] and the references therein. As there is only one action in our unary pPDA Δ, the pLTS $\mathcal{S}(\Delta)$ can be viewed as an infinite-state Markov chain. Let a *run* of Δ be an execution in $\mathcal{S}(\Delta)$ that is either infinite or ending with an empty-stack configuration. If a *run* is finite (i.e., it reaches an empty-stack configuration), we say that it *terminates*. For each configuration $q\alpha$, by $Run(q\alpha)$ we denote the set of runs starting in $q\alpha$. A probability measure $\mathbb{P}$ can be defined over $Run(q\alpha)$ in the standard way, see e.g., [4] for the formal details.

To each configuration $q\alpha$ we associate the random variable $T_{q\alpha} : Run(q\alpha) \to \mathbb{N} \cup \{\infty\}$ that maps each run to the number of its steps, called the *termination time* of the run. For each $i \in \mathbb{N}$ we have that $\mathbb{P}(T_{q\alpha} = i)$, i.e., the probability that a run from $q\alpha$ terminates in i steps, is equal to $\mathcal{P}^{\Delta}_{q\alpha}(a^i)$. From this point of view, language equivalence means equality between the (sub-)distributions of $T_{q_1\alpha_1}$ and $T_{q_2\alpha_2}$. We note that some bounds on $\mathbb{P}(T_{q\alpha} > i)$ were derived in [4], but equivalence seems not to have been analysed so far.

3.1.2. Almost surely terminating pBPA

We first restrict our attention to *almost surely terminating* pBPA $\Delta = (\{q\}, \Gamma, \{a\}, \hookrightarrow)$, i.e., we assume $\mathbb{P}(T_\alpha < \infty) = 1$ for each $\alpha \in \Gamma^*$; for short we write T_α instead of $T_{q\alpha}$ since q is the only control state. Later we extend the proof to all unary pBPA, which together with Theorem 1 (to be proven in Section 4) will complete the proof.

Given $X, Y \in \Gamma$, our (language equivalence) problem is to decide whether the distributions of T_X and T_Y coincide, or whether there is $i \in \mathbb{N}$ such that $\mathbb{P}(T_X = i) \neq \mathbb{P}(T_Y = i)$. To this end it is convenient to use the framework of *generating functions*.

3.1.3. Equation systems for generating functions, with unique solutions

For a random variable T over $\mathbb{N}$ we define $g_T : [0,1] \to [0,1]$ by

$$g_T(z) := \mathbb{E}(z^T) = \sum_{i=0}^{\infty} \mathbb{P}(T = i) \cdot z^i$$

where $\mathbb{E}$ denotes the *expectation* with respect to $\mathbb{P}$.

Using the superscript (i) to denote the i-th derivative, we note that $g_T = g_{T'}$ implies $g_T^{(i)}(0) = g_{T'}^{(i)}(0)$, and thus $\mathbb{P}(T = i) = \mathbb{P}(T' = i)$. The next proposition follows immediately.

Proposition 1. *The distributions of T, T' are the same iff $g_T = g_{T'}$.*

Convention. By $T + T'$ we refer to a random variable with the distribution

$$\mathbb{P}(T + T' = i) \quad = \quad \sum_{j=0}^{i} \mathbb{P}(T = j) \cdot \mathbb{P}(T' = i{-}j).$$

We further assume an almost surely terminating pBPA $\Delta = (\{q\}, \Gamma, \{a\}, \hookrightarrow)$. For $\alpha \in \Gamma^*$ we often write g_α instead of g_{T_α}.

Proposition 2. *For $X, Y \in \Gamma$, the distributions of T_{XY} and $T_X + T_Y$ are the same, and $g_{XY}(z) = g_X(z) \cdot g_Y(z)$.*

Proof. The first part follows by observing that a terminating run from XY naturally corresponds to a terminating run from X followed by a terminating run from Y. For the rest we observe that

$$\begin{aligned}\sum_{i=0}^{\infty} \mathbb{P}(T_X + T_Y = i) \cdot z^i &= \sum_{i=0}^{\infty} \sum_{j=0}^{i} \mathbb{P}(T_X = j) \cdot \mathbb{P}(T_Y = i{-}j) \cdot z^j \cdot z^{i-j} \\ &= (\sum_{i=0}^{\infty} \mathbb{P}(T_X = i) \cdot z^i) \cdot (\sum_{i=0}^{\infty} \mathbb{P}(T_Y = i) \cdot z^i). \qquad \square\end{aligned}$$

We illustrate our equation systems by an example first.

Example 2. *Let us consider a unary pBPA given by $X \overset{0.3}{\hookrightarrow} XY$, $X \overset{0.7}{\hookrightarrow} \varepsilon$, and $Y \overset{0.6}{\hookrightarrow} X$, $Y \overset{0.4}{\hookrightarrow} \varepsilon$. We can easily check that*

$$\begin{aligned}g_X(z) = \mathbb{E}(z^{T_X}) = 0.3 \cdot \mathbb{E}(z^{T_X} \mid \textit{rule } X \overset{0.3}{\hookrightarrow} XY \textit{ is taken in the first step}) + \\ + 0.7 \cdot \mathbb{E}(z^{T_X} \mid \textit{rule } X \overset{0.7}{\hookrightarrow} \varepsilon \textit{ is taken in the first step}) = \\ = 0.3 \cdot \mathbb{E}(z^{1+T_X+T_Y}) + 0.7 \cdot \mathbb{E}(z^1) = z \cdot (0.3 \cdot g_X(z) \cdot g_Y(z) + 0.7)\,.\end{aligned}$$

Similarly we can derive $g_Y(z) = z \cdot (0.6 \cdot g_X(z) + 0.4)$.

For any $z \in [0,1]$*, the pair* $(g_X(z), g_Y(z))$ *is thus a fixed point of the function* $z \cdot \boldsymbol{f}$ *where* $\boldsymbol{f}(x_1, x_2) = (0.3x_1x_2 + 0.7, 0.6x_1 + 0.4)$.

Generally, for our assumed almost surely terminating pBPA $\Delta = (\{q\}, \Gamma, \{a\}, \hookrightarrow)$ we define the quadratic function $\boldsymbol{f} : [0,1]^\Gamma \to [0,1]^\Gamma$ by

$$\boldsymbol{f}_X(\boldsymbol{x}) := \sum_{X \overset{p}{\hookrightarrow} YZ} p\, \boldsymbol{x}_Y \boldsymbol{x}_Z + \sum_{X \overset{p}{\hookrightarrow} Y} p\, \boldsymbol{x}_Y + \sum_{X \overset{p}{\hookrightarrow} \varepsilon} p \qquad \text{for } \boldsymbol{x} \in [0,1]^\Gamma. \quad (1)$$

Reasoning as in Example 2, we note that the vector $\boldsymbol{g}(z) := (g_X(z))_{X \in \Gamma}$ satisfies $\boldsymbol{g}(z) = z \cdot \boldsymbol{f}(\boldsymbol{g}(z))$ for each $z \in [0,1]$, i.e., $\boldsymbol{g}(z)$ is a fixed point of $z \cdot \boldsymbol{f}$. Using Proposition 4 we will show that the fixed point is unique. The proof refers to several results in the literature. We first sketch one elementary fact separately, after recalling the notion of Jacobian matrices.

Consider a function $\boldsymbol{F} : \mathbb{R}^k \to \mathbb{R}^k$, i.e., a k-tuple $(F_1, F_2, \ldots, F_k)$ with $F_i : \mathbb{R}^k \to \mathbb{R}$. Assume moreover that the partial derivatives of each F_i exist throughout $\mathbb{R}^k$. For $\boldsymbol{x} = (x_1, x_2, \ldots, x_k)$, we denote by $\boldsymbol{F}'(\boldsymbol{x})$ the Jacobian matrix of $\boldsymbol{F}(\boldsymbol{x})$, i.e., the $(k \times k)$-matrix with $\boldsymbol{F}'_{ij}(\boldsymbol{x}) := \frac{\partial}{\partial \boldsymbol{x}_j} \boldsymbol{F}_i(\boldsymbol{x})$.

Proposition 3. *Let* $\boldsymbol{F} : \mathbb{R}^k \to \mathbb{R}^k$ *be a quadratic function (hence* $F_i(x_1, \ldots, x_k)$ *is a sum of terms of the type* cx_jx_ℓ*,* cx_j*, or* c*, where* $c \in \mathbb{R}$*). For* $\boldsymbol{u}, \boldsymbol{v} \in \mathbb{R}^k$ *we have* $\boldsymbol{F}(\boldsymbol{u}) - \boldsymbol{F}(\boldsymbol{v}) = \boldsymbol{F}'((\boldsymbol{u} + \boldsymbol{v})/2)(\boldsymbol{u} - \boldsymbol{v})$.

Proof. We can write $F_i(\boldsymbol{x}) = \boldsymbol{x}^T B \boldsymbol{x} + \boldsymbol{\ell}\boldsymbol{x} + c$, where we view $\boldsymbol{x}$ as a column vector, $\boldsymbol{x}^T$ is the transpose of $\boldsymbol{x}$, B is a symmetric real $k \times k$-matrix, $\boldsymbol{\ell} \in \mathbb{R}^k$ is a row vector, and $c \in \mathbb{R}$.

By symmetry of B we get $F'_i(\boldsymbol{x}) = 2\boldsymbol{x}^T B + \boldsymbol{\ell}$, and thus
$F_i(\boldsymbol{u}) - F_i(\boldsymbol{v}) = \boldsymbol{u}^T B \boldsymbol{u} - \boldsymbol{v}^T B \boldsymbol{v} + \boldsymbol{\ell}(\boldsymbol{u} - \boldsymbol{v}) = (\boldsymbol{u} + \boldsymbol{v})^T B(\boldsymbol{u} - \boldsymbol{v}) + \boldsymbol{\ell}(\boldsymbol{u} - \boldsymbol{v}) =$
$F'_i((\boldsymbol{u} + \boldsymbol{v})/2)(\boldsymbol{u} - \boldsymbol{v})$. □

Proposition 4. *For each* $z \in [0,1]$*, the function* $z \cdot \boldsymbol{f} : [0,1]^\Gamma \to [0,1]^\Gamma$ *has a unique fixed point in* $[0,1]^\Gamma$*. (By* $\boldsymbol{f}$ *we refer to the function (1), defined for* Δ*.)*

Proof. We first handle the case $z = 1$. It is known (see [9, 11]) that the least nonnegative fixed point of $\boldsymbol{f}$ is equal to the vector of termination probabilities (of runs starting from $X \in \Gamma$), which is $\mathbf{1} := (1, 1, \ldots, 1)$ by our assumption. Hence there is no other fixed point of $\boldsymbol{f}$ in $[0,1]^\Gamma$.

By $\boldsymbol{f}'(\boldsymbol{x})$ we denote the Jacobian matrix of $\boldsymbol{f}(\boldsymbol{x})$, i.e., the $\Gamma \times \Gamma$-matrix with $\boldsymbol{f}'_{XY}(\boldsymbol{x}) := \frac{\partial}{\partial \boldsymbol{x}_Y} \boldsymbol{f}_X(\boldsymbol{x})$. We claim that the spectral radius of (the real-valued nonnegative square matrix) $\boldsymbol{f}'(\mathbf{1})$ is at most one. Towards a contradiction, suppose that $\boldsymbol{f}'(\mathbf{1})$ is greater than one. It follows from a fact about nonnegative matrices, see [3, Corollary 2.1.6], that $\boldsymbol{f}'(\mathbf{1})$ has an *irreducible principal submatrix* M with the same spectral radius. The matrix M being a *principal submatrix* means that M is a $\Gamma' \times \Gamma'$-matrix with $M_{XY} = \boldsymbol{f}'_{XY}(\mathbf{1})$ for all $X, Y \in \Gamma'$, where Γ' is a subset of Γ. The matrix M being *irreducible* means that the graph that has Γ' as the set of vertices and $\{(X,Y) \mid X, Y \in \Gamma', M_{XY} > 0\}$ as the set of edges is strongly connected.

As the spectral radius of M is greater than one, it follows from the correctness of the algorithm in [11, Figure 8] that $\mathbb{P}(T_X < \infty) < 1$ for some $X \in \Gamma'$. This contradicts our assumption that Δ is almost surely terminating. So the spectral radius of $\boldsymbol{f}'(\mathbf{1})$ is indeed at most one.

We now fix $z < 1$ and define $\boldsymbol{F}(\boldsymbol{x}) := z \cdot \boldsymbol{f}(\boldsymbol{x})$ (for $\boldsymbol{x} \in [0,1]^\Gamma$); by $\boldsymbol{F}'(\boldsymbol{x})$ we denote the Jacobian matrix of $\boldsymbol{F}(\boldsymbol{x})$. Then the spectral radius of $\boldsymbol{F}'(\mathbf{1}) = z \cdot \boldsymbol{f}'(\mathbf{1})$ is strictly less than one. Although the spectral radius itself is not a norm, it follows from Theorem 5.3.5 of [20] that there is a vector norm $\|\cdot\| : \mathbb{R}^\Gamma \to \mathbb{R}$ whose induced matrix norm satisfies $\left\|\boldsymbol{F}'(\mathbf{1})\right\| < 1$. Since $\boldsymbol{F}$ is a quadratic function, by Proposition 3 we have $\boldsymbol{F}(\boldsymbol{u}) - \boldsymbol{F}(\boldsymbol{v}) = \boldsymbol{F}'((\boldsymbol{u}+\boldsymbol{v})/2)(\boldsymbol{u}-\boldsymbol{v})$.

Considering the mentioned norm, for all $\boldsymbol{u}, \boldsymbol{v} \in [0,1]^\Gamma$ we have

$$\begin{aligned}\|\boldsymbol{F}(\boldsymbol{u}) - \boldsymbol{F}(\boldsymbol{v})\| = \left\|\boldsymbol{F}'((\boldsymbol{u}+\boldsymbol{v})/2)(\boldsymbol{u}-\boldsymbol{v})\right\| \leq \left\|\boldsymbol{F}'((\boldsymbol{u}+\boldsymbol{v})/2)\right\| \|\boldsymbol{u}-\boldsymbol{v}\| \leq \\ \leq \underbrace{\left\|\boldsymbol{F}'(\mathbf{1})\right\|}_{<1} \|\boldsymbol{u}-\boldsymbol{v}\| \ ,\end{aligned}$$

where the last inequality holds since all coefficients in $\boldsymbol{F}'(\boldsymbol{x})$ are nonnegative. The Banach fixed-point theorem now implies that $\boldsymbol{F}$ has a unique fixed point in $[0,1]^\Gamma$. □

3.1.4. Completing the proof of Lemma 1

We have shown that the question if $X_1, X_2 \in \Gamma$ are *not* equivalent (in our assumed almost surely terminating pBPA Δ) reduces to the question if the closed formula

$$\exists z \in [0,1]\, \exists \boldsymbol{x} \in [0,1]^\Gamma \, (\boldsymbol{x} = z \cdot \boldsymbol{f}(\boldsymbol{x}) \ \wedge \ \boldsymbol{x}_{X_1} \neq \boldsymbol{x}_{X_2})$$

is true in $ExTh(\mathbb{R})$. The latter condition is decidable in polynomial space, see [6, 22].

Now we show how to generalize the claim of Lemma 1 to pBPA in which the probability of nontermination can be positive. Let us fix a (general) unary pBPA $\Delta = (\{q\}, \Gamma, \{a\}, \hookrightarrow)$. Given X_1, $X_2 \in \Gamma$, our problem is to decide whether the distributions of T_{X_1} and T_{X_2} coincide. We solve this by a reduction to the case of almost surely terminating unary pBPA; using a result from [4], we construct an almost surely terminating pBPA $\Delta_\bullet$ whose distribution of termination time is closely related to the distribution in Δ.

For the definition of $\Delta_\bullet$, the notion of *termination probabilities* is crucial: For $\alpha \in \Gamma^*$ we write $[\alpha] := \mathbb{P}(T_\alpha < \infty)$, i.e., $[\alpha]$ is the probability that a run starting in α terminates. Note that for $X, Y \in \Gamma$ we have $[XY] = [X] \cdot [Y]$. Let $\mathit{Term} := \{X \in \Gamma \mid [X] > 0\}$. The set Term can be computed in polynomial time by a standard (qualitative) reachability analysis. We now define $\Delta_\bullet$ as the unary pBPA with the stack alphabet $\Gamma_\bullet := \mathit{Term}$ where the transition rules $\hookrightarrow_\bullet$ are induced as follows: if $X \overset{x}{\hookrightarrow} \alpha$ and $\alpha \in \Gamma_\bullet^{\leq 2}$ (hence also $X \in \Gamma_\bullet$) then $X \xrightarrow{x \cdot [\alpha]/[X]}_\bullet \alpha$. We use $T_\alpha^\bullet$ for the random variable that maps the runs of $\Delta_\bullet$ starting in $\alpha \in \Gamma_\bullet^*$ to their termination time. We have the following proposition from [4]:

Proposition 5 (Proposition 6 of [4]). *The pBPA $\Delta_\bullet$ is almost surely terminating, i.e., we have $\mathbb{P}(T^\bullet_\alpha < \infty) = 1$ for all configurations $\alpha \in \Gamma^*_\bullet$. Moreover, for all $X \in \Gamma_\bullet$ and all $i \in \mathbb{N}$ we have that $\mathbb{P}(T_X = i \mid \varepsilon \text{ is reached}) = \mathbb{P}(T^\bullet_X = i)$, where the left-hand side refers to Δ and the right-hand side to $\Delta_\bullet$.*

It follows that for all $X \in \Gamma$ and all $i \in \mathbb{N}$ we have $\mathbb{P}(T_X = i) = [X] \cdot \mathbb{P}(T^\bullet_X = i)$, and hence for the generating functions we get $g_{T_X}(z) := \mathbb{E}\left(z^{T_X}\right) = [X] \cdot g_{T^\bullet_X}(z)$. Let $\boldsymbol{f}_\bullet : [0,1]^{\Gamma\bullet} \to [0,1]^{\Gamma\bullet}$ be the quadratic function defined as $\boldsymbol{f}$ in (1), but now for $\Delta_\bullet$. By similar reasoning as before it follows that two configurations $X_1, X_2 \in \Gamma$ are *not* equivalent if and only if the closed formula

$$\exists z \in [0,1]\, \exists \boldsymbol{x} \in [0,1]^{\Gamma\bullet} \Big(\boldsymbol{x} = z \cdot \boldsymbol{f}_\bullet(\boldsymbol{x}) \ \wedge [X_1] \cdot \boldsymbol{x}_{X_1} \neq [X_2] \cdot \boldsymbol{x}_{X_2} \Big) \qquad (2)$$

is true in $ExTh(\mathbb{R})$. The termination probabilities are in general irrational. But by Theorem 18 of [12] (see also the proof of Theorem 21 of [12])[1] we have the following proposition:

Proposition 6. *Let $\boldsymbol{t} \in (0,1]^{Term}$ be the vector of termination probabilities, i.e., $\boldsymbol{t} := ([X] \mid X \in Term)$. Let $\boldsymbol{x} = (\boldsymbol{x}_X \mid X \in Term)$ be a vector of variables. One can compute in polynomial space a formula $\Phi(\boldsymbol{x}) \in ExTh(\mathbb{R})$ of polynomial length and with free variables $\boldsymbol{x}$ so that for all $\boldsymbol{u} \in \mathbb{R}^{Term}$ we have that $\Phi(\boldsymbol{u})$ holds if and only if $\boldsymbol{u} = \boldsymbol{t}$.*

Using Proposition 6 one can express the termination probabilities and the coefficients of the function $\boldsymbol{f}_\bullet$ in $ExTh(\mathbb{R})$. Hence a polynomial space computation rephrases (2) as a closed $ExTh(\mathbb{R})$-formula of polynomial length. By appealing again to decision procedures for $ExTh(\mathbb{R})$ [6, 22], we have established the claim of Lemma 1 for all unary pBPA. Theorem 1 (to be proven in Section 4) thus completes a proof of Lemma 1, and also of Point 1. in Theorem 2.

3.2. Hardness for Polynomial Identity Testing

In this section we prove the hardness result expressed in Point 2. in Theorem 2. We recall from Section 2 that we can view the ACIT problem as the decision problem that asks if two variable-free circuits with the operations $+, *$ output the same number. After we prove Lemma 2 below and Theorem 1, Point 2. in Theorem 2 will be established.

Lemma 2. *ACIT is polynomially reducible to the language equivalence problem for unary pPDA.*

Proof. Our proof closely follows a proof given in [17] for (non-unary) probabilistic *visibly* pushdown automata.

Let C and C' be two circuits where each non-input gate is labelled with $+$ or $*$. By the *depth* of a gate we mean its distance from the output gate. We

[1] In fact, in [12] the formalism of Recursive Markov Chains (RMCs) is used. RMCs and pPDAs (with unary alphabet) are equivalent in a precise sense, see also [12].

assume that the circuits have the following form (to which they can be safely transformed in polynomial time): the inputs of a depth-i gate both have depth $i{+}1$, $+$-gates have even depths, $*$-gates have odd depths, and the input gates all have the same depth d. Thus in each circuit any path from an input gate to the output gate has length d. Let C have the gates $g_1, \ldots, g_m$, with g_1 being the output gate, and let C' have the gates $g_{m+1}, \ldots, g_n$, with g_{m+1} being the output gate.

We define a unary pPDA $\Delta = (Q, \Gamma, \{a\}, \hookrightarrow)$ with $Q = \{q_0, q_1, \ldots, q_n\}$ and $\Gamma = \{X_0, X_1, \ldots, X_n\}$; the rules $\hookrightarrow$ are constructed as follows:

- For each $+$-gate $g_i = g_j + g_k$ we include a transition $q_i X \xhookrightarrow{1/2} q_j X$ and $q_i X \xhookrightarrow{1/2} q_k X$ for all $X \in \Gamma$.
- For each $*$-gate $g_i = g_j * g_k$ we include a transition $q_i X \xhookrightarrow{1} q_j X_k X$ for all $X \in \Gamma$.
- For each 1-labelled input gate g_i we include a transition $q_i X_j \xhookrightarrow{1} q_j$ for all $j \in \{0, \ldots, n\}$.

(The 0-labelled input gates do not give rise to any transitions.)

We will show that each of $q_1 X_0$, $q_{m+1} X_0$ accepts at most one word with positive probability, and that $q_1 X_0$ and $q_{m+1} X_0$ are equivalent if and only if C and C' output the same number.

To this end, we define a sequence of words $w_0, w_1, \ldots, w_d \in \{a\}^*$ in reversed order as follows: $w_d = a$; if $e < d$ is even then $w_e = a w_{e+1}$; if $e < d$ is odd then $w_e = a w_{e+1} w_{e+1}$. We also define the numbers $M_0, M_1, \ldots, M_d$ as follows: $M_d = 1$; if $e < d$ is even then $M_e = 2M_{e+1}$; if $e < d$ is odd then $M_e = (M_{e+1})^2$.

Claim. Let $i \in \{1, \ldots, n\}$ and $X \in \Gamma$. Let e denote the depth of gate g_i and $N_i \in \mathbb{N}$ the output number of g_i. Then we have $\mathcal{P}^{\Delta}_{q_i X}(w_e) = N_i / M_e$. Moreover, $\mathcal{P}^{\Delta}_{q_i X}(w) = 0$ for all $w \neq w_e$.

The claim obviously implies that $q_1 X_0$ and $q_{m+1} X_0$ are language equivalent (accepting w_0 with the same probability) iff C, C' output the same number (by g_1 and g_{m+1}). Now we prove the claim, by which the proof of the lemma will be finished. We first observe that

$$\text{if the pLTS } \mathcal{S}(\Delta) \text{ has a path from } q_i X_j \text{ to } q_k, \text{ then } j = k\,. \tag{3}$$

We prove *Claim* by (reverse) induction on the depth e of g_i. If $e = d$ then g_i is an input gate; if labelled with 1 then $\mathcal{P}^{\Delta}_{q_i X}(w_d) = 1 = N_i / M_d$ and $\mathcal{P}^{\Delta}_{q_i X}(w) = 0$ for all $w \in \{a\}^* \setminus \{w_d\}$; if labelled with 0 then $\mathcal{P}^{\Delta}_{q_i X}(w) = 0 = N_i / M_d$ for all $w \in \{a\}^*$. Let us now assume $e < d$.

- Let $g_i := g_j + g_k$ be a gate of (even) depth e. Then we have:

$$\begin{aligned}
\mathcal{P}^{\Delta}_{q_iX}(w_e) &= \frac{1}{2}\mathcal{P}^{\Delta}_{q_jX}(w_{e+1}) + \frac{1}{2}\mathcal{P}^{\Delta}_{q_kX}(w_{e+1}) && \text{(by the rules of } \Delta\text{)} \\
&= \frac{1}{2}\cdot\frac{N_j}{M_{e+1}} + \frac{1}{2}\cdot\frac{N_k}{M_{e+1}} && \text{(by the induction hypothesis)} \\
&= N_i/M_e && \text{(by definition)}
\end{aligned}$$

  We also have $\mathcal{P}^{\Delta}_{q_iX}(\varepsilon) = 0$ and $\mathcal{P}^{\Delta}_{q_iX}(av) = \frac{1}{2}\mathcal{P}^{\Delta}_{q_jX}(v) + \frac{1}{2}\mathcal{P}^{\Delta}_{q_kX}(v) = 0$ if $v \neq w_{e+1}$ (and thus $av \neq w_e$); here we also use the induction hypothesis.

- Let $g_i := g_j * g_k$ be a gate of (odd) depth e. Then we have:

$$\begin{aligned}
\mathcal{P}^{\Delta}_{q_iX}(w_e) &= \mathcal{P}^{\Delta}_{q_jX_kX}(w_{e+1}w_{e+1}) && \text{(by the rules of } \Delta\text{)} \\
&= \mathcal{P}^{\Delta}_{q_jX_k}(w_{e+1}) \cdot \mathcal{P}^{\Delta}_{q_kX}(w_{e+1}) && \text{(by the observation (3))} \\
&= \frac{N_j}{M_{e+1}} \cdot \frac{N_k}{M_{e+1}} && \text{(by the induction hypothesis)} \\
&= N_i/M_e && \text{(by definition)}
\end{aligned}$$

  We have $\mathcal{P}^{\Delta}_{q_iX}(\varepsilon) = 0$; moreover, $\mathcal{P}^{\Delta}_{q_iX}(av) > 0$ implies that $v = w_{e+1}w_{e+1}$, by the induction hypothesis. □

4. Proof of Theorem 1

In this section we prove Theorem 1, for convenience restated here; the proofs are shown by suitable modifications of standard constructions from the literature.

Theorem 1. *The following problems are interreducible in polynomial time, without changing the input alphabet or terminal alphabet, respectively:*

1. *language equivalence for pPDA;*
2. *language equivalence for pBPA;*
3. *multiplicity equivalence for (ε-free) finite-multiplicity grammars.*

The direction 2.⇒1. is trivial. We prove 1.⇒3. in Section 4.2, and 3.⇒2. in Section 4.3, but first we define the following convenient generalisation of grammars.

4.1. Multigrammars

A *multigrammar*[2] is a quadruple (V, Σ, R, S) like a grammar, but we have $R \subseteq V \times \mathbb{N}_+ \times (V \cup \Sigma)^+$, where $\mathbb{N}_+ = \{1, 2, \dots\}$. A grammar is the special case with $R \subseteq V \times \{1\} \times (V \cup \Sigma)^+$. We write $X \xrightarrow{k} \alpha$ if $(X, k, \alpha) \in R$, and

[2] An equivalent formalism would involve algebraic formal power series, but we wish to emphasise the proximity to grammars, which is a standard concept.

we call k *the multiplicity of the rule*. Multigrammars inherit the notions of derivations etc. from grammars. The multiplicity of a derivation is the product of the multiplicities of the applied rules. Finally, the multiplicity $m_G(w)$ of a word $w \in \Sigma^*$ in a multigrammar G is defined as the sum of the multiplicities of all leftmost derivations of w in G. (This is consistent with the definition given for grammars.) When discussing complexity questions, we assume that the multiplicities of the rules are given in binary.

Lemma 3. *A multigrammar $G = (V, \Sigma, R, S)$ can be transformed to a grammar $G' = (V', \Sigma, R', S)$ in polynomial time so that for all words $w \in \Sigma^*$ we have $m_G(w) = m_{G'}(w)$.*

Proof. If $X \xrightarrow{2\ell} \alpha$ is a rule of G, where $\ell \in \mathbb{N}_+$, then we add fresh nonterminals, say A, A', remove the rule $X \xrightarrow{2\ell} \alpha$, and add the rules $X \xrightarrow{1} A$, $X \xrightarrow{1} A'$, $A' \xrightarrow{1} A$, and $A \xrightarrow{\ell} \alpha$. If $X \xrightarrow{2\ell+1} \alpha$ is a rule of G then we proceed similarly but we also add the rule $X \xrightarrow{1} \alpha$. We repeat this construction until the multiplicity of each rule is 1. □

4.2. Reduction from pPDA to Grammars (1.⇒3. in Theorem 1)

Let us consider a (non-blocking) pPDA $\Delta = (Q, \Gamma, \Sigma, \hookrightarrow)$ and two configurations p_1X_1 and p_2X_2. We aim to construct two multigrammars G_1, G_2 so that p_1X_1 and p_2X_2 are equivalent in Δ iff G_1 and G_2 are multiplicity equivalent.

We assume that there is a control state $q_\perp$ such that whenever $q\varepsilon$ (an empty-stack configuration) is reached from p_1X_1 or p_2X_2 then $q = q_\perp$. This could be easily achieved by considering the initial configurations $\bar{p}_1X_1$ and $\bar{p}_2X_2$ for newly added control states $\bar{p}_1, \bar{p}_2$, when we add a fresh stack symbol $\perp$ and transitions $\bar{p}_1X_1 \overset{a,1}{\hookrightarrow} p_1X_1\perp$, $\bar{p}_2X_2 \overset{a,1}{\hookrightarrow} p_2X_2\perp$, $t\perp \overset{a,1}{\hookrightarrow} q_\perp$ for all $t \in Q$ and $a \in \Sigma$. We thus do not lose generality by the above assumption.

Let $M \in \mathbb{N}$ be the smallest number such that all probabilities in $\hookrightarrow$ are integer multiples of $1/M$; the binary representation of M is polynomial in the size of Δ. We now construct multigrammars $G_1 = (V, \Sigma, R, \langle p_1X_1q_\perp\rangle)$ and $G_2 = (V, \Sigma, R, \langle p_2X_2q_\perp\rangle)$ with $V := \{\langle pXq\rangle \mid p, q \in Q,\ X \in \Gamma\}$ and with the rules constructed as follows:

- for every $pX \overset{a,x}{\hookrightarrow} qYZ$ and every $r, s \in Q$ we include the multigrammar rule $\langle pXs\rangle \xrightarrow{xM} a\langle qYr\rangle\langle rZs\rangle$;
- for every $pX \overset{a,x}{\hookrightarrow} rZ$ and every $s \in Q$ we include $\langle pXs\rangle \xrightarrow{xM} a\langle rZs\rangle$;
- for every $pX \overset{a,x}{\hookrightarrow} s$ we include $\langle pXs\rangle \xrightarrow{xM} a$.

For any $w \in \Sigma^*$ and $\langle pXq\rangle \in V$, each leftmost derivation $\langle pXq\rangle \Rightarrow \cdots \Rightarrow w$ with multiplicity $x \cdot M^{|w|}$ corresponds to an execution in $\mathcal{S}(\Delta)$ going from pX to q on w that has probability x, and *vice versa*. Hence for all $w \in \Sigma^*$ we have $m_{G_1}(w) = M^{|w|} \cdot \mathcal{P}^{\Delta}_{p_1X_1}(w)$ and $m_{G_2}(w) = M^{|w|} \cdot \mathcal{P}^{\Delta}_{p_2X_2}(w)$. Therefore p_1X_1 and p_2X_2 are equivalent in Δ iff G_1 and G_2 are multiplicity equivalent. The

multigrammars G_1, G_2 can be transformed to grammars by Lemma 3; we have thus proven the direction 1.⇒3. in Theorem 1.

4.3. Reduction from Grammars to pBPA (3.⇒2. in Theorem 1)

A multigrammar $G = (V, \Sigma, R, S)$ is said to be in *Greibach normal form (GNF)* if each rule is of the form $X \xrightarrow{k} a\beta$ where $a \in \Sigma$ and $\beta \in V^*$; if, moreover, $|\beta| \leq 2$ for each rule then G is in *2-GNF*. The following lemma is crucial for our proof; recall that we have restricted ourselves to ε-free (multi)grammars.

Lemma 4. *A multigrammar $G = (V, \Sigma, R, S)$ with finite multiplicity can be transformed in polynomial time into a multigrammar $G' = (V', \Sigma, R', S)$ in 2-GNF so that for all $w \in \Sigma^*$ we have $m_G(w) = m_{G'}(w)$.*

Proof. We first note that a standard conversion of grammars to GNF (as given, e.g., in [16], pp. 277–279) is exponential. We thus modify a construction suggested in [23] and later presented differently in [25, 1]. Care is needed to ensure correct manipulation of the multiplicities of the rules.

Let $G = (V, \Sigma, R, S)$ be a multigrammar with finite multiplicity. Without loss of generality we assume that each nonterminal can derive a terminal word, and that each nonterminal is reachable from S, i.e., it occurs in some $\alpha \in (V \cup \Sigma)^+$ derivable from S. (A standard procedure removing the rules with "redundant" nonterminals that do not satisfy these conditions does not affect the multiplicities $m_G(w)$.) We first eliminate the "unit rules", i.e., the rules of the form $X \xrightarrow{k} Y$ with $X, Y \in V$. If there is such a rule then there must be a rule $X \xrightarrow{k} Y$ such that there is no unit rule with Y on the left-hand side. (Otherwise there would be a cycle of unit rules, we would have $X \Rightarrow \cdots \Rightarrow X$, contradicting the assumption of finite multiplicity of G.) Let $Y \xrightarrow{\ell_1} \alpha_1, \ldots, Y \xrightarrow{\ell_j} \alpha_j$ (where $j > 0$) be all the rules with Y on the left-hand side. We remove $X \xrightarrow{k} Y$ and add the rules $X \xrightarrow{k\ell_1} \alpha_1, \ldots, X \xrightarrow{k\ell_j} \alpha_j$. By "adding the rules" we mean possibly increasing their multiplicities: it might happen that a rule $X \xrightarrow{m} \alpha_i$ has been already present in the multigrammar; in this case it is replaced with $X \xrightarrow{m+k\ell_i} \alpha_i$. This transformation obviously keeps all $m_G(w)$ unchanged. We thus further assume that G has no unit rules.

We now transform the multigrammar $G = (V, \Sigma, R, S)$ to a multiplicity-equivalent multigrammar $G_\bullet = (V_\bullet, \Sigma, R_\bullet, S)$ where $V_\bullet := V \cup \{B_{XY} \mid X, Y \in V\}$ for fresh nonterminals B_{XY}. The set $R_\bullet$ is constructed as follows:

- each rule in R of the form $X \xrightarrow{k} a\alpha$ where $a \in \Sigma$ is included in $R_\bullet$;
- if $X \xrightarrow{k} a\alpha$ is in R, and $a \in \Sigma$, then $Y \xrightarrow{k} a\alpha B_{XY}$ is included in $R_\bullet$, for each $Y \in V$;
- if $Y \xrightarrow{k} X\beta$ is in R then $B_{XY} \xrightarrow{k} \beta$ is included in $R_\bullet$;
- if $Z \xrightarrow{k} X\gamma$ is in R, then $B_{XY} \xrightarrow{k} \gamma B_{ZY}$ is included in $R_\bullet$, for each $Y \in V$.

We note that each rule $Z \xrightarrow{k} \delta$ in $R_\bullet$ satisfies the following: if $Z \in V$ then $\delta = a\delta'$ for some $a \in \Sigma$, and if $Z \in V_\bullet \setminus V$ then $\delta = x\delta'$ for some $x \in (\Sigma \cup V)$.

We illustrate the crux of the fact that G and $G_\bullet$ are multiplicity equivalent by an example. Suppose a leftmost derivation of a word $w \in \Sigma^*$ in G, namely

$$S \Rightarrow \cdots \Rightarrow uY\delta \Rightarrow uZ\gamma\delta \Rightarrow uX\beta\gamma\delta \Rightarrow ua\alpha\beta\gamma\delta \Rightarrow \cdots \Rightarrow w \tag{4}$$

where the depicted segment uses the rules $Y \xrightarrow{k_1} Z\gamma$, $Z \xrightarrow{k_2} X\beta$, $X \xrightarrow{k_3} a\alpha$.

The segment can be written in $G_\bullet$ as

$$uY\delta \Rightarrow ua\alpha B_{XY}\delta \Rightarrow ua\alpha\beta B_{ZY}\delta \Rightarrow ua\alpha\beta\gamma\delta$$

using the rules $Y \xrightarrow{k_3} a\alpha B_{XY}$, $B_{XY} \xrightarrow{k_2} \beta B_{ZY}$, $B_{ZY} \xrightarrow{k_1} \gamma$. It is clear that any leftmost derivation $S \Rightarrow \cdots \Rightarrow w$ in G is naturally composed of segments as depicted in (4); each segment uses rules in which the right-hand side starts with a nonterminal except the last rule in which the right-hand side starts with a terminal. We can thus easily check that each leftmost derivation of w in G has a corresponding derivation in $G_\bullet$, with the same multiplicity; the latter derivation is leftmost except that rewriting $B \in (V_\bullet \setminus V)$ is always performed preferentially. On the other hand, each such derivation in $G_\bullet$ (a leftmost derivation of w modified so that $B \in (V_\bullet \setminus V)$ is always rewritten preferentially) has a corresponding leftmost derivation in G, with the same multiplicity. This makes clear that G and $G_\bullet$ are multiplicity equivalent.

The multigrammar $G_\bullet$ may still contain rules whose right-hand sides start with a nonterminal; these rules are of the form $B_{XY} \xrightarrow{k} Z\gamma$ where $Z \in V$. Each such rule can be removed and replaced with the rules $B_{XY} \xrightarrow{k\ell_1} \gamma_1\gamma$, $\ldots$, $B_{XY} \xrightarrow{k\ell_n} \gamma_n\gamma$ where $Z \xrightarrow{\ell_1} \gamma_1$, $\ldots$, $Z \xrightarrow{\ell_n} \gamma_n$ are all rules with Z on the left-hand side; we recall that each γ_i starts with a terminal. This transformation preserves the multiplicities.

It is now clear that it suffices to handle the case in which all rules in $G = (V, \Sigma, R, S)$ have the right-hand sides starting with a terminal. For each $a \in \Sigma$ we add a fresh nonterminal N_a and the rule $N_a \xrightarrow{1} a$, and in each right-hand side of the form $b\alpha$, for $b \in \Sigma$, we replace each occurrence of a in α with N_a. The resulting grammar, for convenience again described as $G = (V, \Sigma, R, S)$, is in GNF.

It remains to achieve the 2-GNF restrictions. For each rule $r = (X \xrightarrow{k} aY_1 \cdots Y_m)$ with $m \geq 2$ we introduce fresh nonterminals $F_1^r, \ldots, F_{m-1}^r$, remove the rule r and add the rules $X \xrightarrow{k} aF_1^r$, $F_1^r \xrightarrow{1} Y_1F_2^r$, $\ldots$, $F_{m-1}^r \xrightarrow{1} Y_{m-1}Y_m$. This transformation also preserves the multiplicities. Let $\mathcal{E}$ denote the set of all new nonterminals (of the form F_i^r). For each rule $F \xrightarrow{1} YZ$ with $F \in \mathcal{E}$ we have $Y \in V$, and for all rules $X \xrightarrow{k} \gamma$ with $X \in V$ we have that $\gamma = a$ or $\gamma = aZ$ where $a \in \Sigma$ and $Z \in V \cup \mathcal{E}$. Now for each rule $F \xrightarrow{1} YZ$ with $F \in \mathcal{E}$ we do the following: we remove the rule, and add the rules $F \xrightarrow{k_1} \gamma_1 Z$, $\ldots$, $F \xrightarrow{k_\ell} \gamma_\ell Z$, where $Y \xrightarrow{k_1} \gamma_1$, $\ldots$, $Y \xrightarrow{k_\ell} \gamma_\ell$ are all rules with Y on the left-hand side. This

transformation preserves the multiplicities and results in a multigrammar in 2-GNF. □

Completing 3.⇒2. in Theorem 1.

Lemma 4 allows us to start with multigrammars $G_1 = (V, \Sigma, R, S_1)$ and $G_2 = (V, \Sigma, R, S_2)$ in 2-GNF; the unification of the sets V and R, if different in G_1 and G_2, is achieved by taking the disjoint union. For $X \in V$ and $a \in \Sigma$ we put $d_{X,a} := \sum_{X \xrightarrow{k} a\beta} k$; then we define $p := 1/\max_{X,a} d_{X,a}$.

We now define the pBPA $\Delta = (\{q\}, V \cup \{\bot\}, \Sigma, \hookrightarrow)$, where $\bot \notin V \cup \Sigma$ and the rules of Δ are constructed as follows:

- for each $(X \xrightarrow{k} a\beta) \in R$ we add $X \xhookrightarrow{a,kp} \beta$;
- for all $X \in V$ and $a \in \Sigma$ with $d_{X,a}p < 1$, we add $X \xhookrightarrow{a,1-d_{X,a}p} \bot$;
- for each $a \in \Sigma$ we add $\bot \xhookrightarrow{a,1} \bot$.

We observe that the pBPA Δ, starting with X, simulates exactly the leftmost derivations of the grammar starting with X. So we have for all $w \in \Sigma^*$ that $\mathcal{P}^{\Delta}_{S_1}(w) = m_{G_1}(w) \cdot p^{|w|}$ and $\mathcal{P}^{\Delta}_{S_2}(w) = m_{G_2}(w) \cdot p^{|w|}$. Hence G_1 and G_2 are multiplicity equivalent if and only if S_1 and S_2 are equivalent in Δ. This completes the proof of Theorem 1.

5. Conclusions

We have studied the language equivalence problem for probabilistic pushdown automata, showing that the problem is interreducible with the problem of multiplicity equivalence for context-free grammars whose decidability is a long-standing open question. We have also provided complexity bounds for the restriction of the problem to a unary input alphabet.

Our results indicate that significant effort will have to be put into resolving the decidability status of the language equivalence problem. Another possibility for further work is to try to extend our results to probabilistic pushdown automata with ε-steps.

Acknowledgements. Vojtěch Forejt is also affiliated with Masaryk University, Czech Republic. Petr Jančar has been supported by the Grant Agency of the Czech Rep., project GAČR:P202/11/0340. Stefan Kiefer is supported by a Royal Society University Research Fellowship.

References

[1] S. P. Abney, D. A. McAllester, and F. Pereira. Relating probabilistic grammars and automata. In *ACL*, pages 542–549. ACL, 1999.

[2] E.E. Allender, P. Bürgisser, J. Kjeldgaard-Pedersen, and P. Bro Miltersen. On the complexity of numerical analysis. *SIAM J. Comput.*, 38(5):1987–2006, 2009.

[3] A. Berman and R. J. Plemmons. *Nonnegative Matrices in the Mathematical Sciences*. SIAM, 1994.

[4] T. Brázdil, S. Kiefer, A. Kučera, and I.H. Vařeková. Runtime analysis of probabilistic programs with unbounded recursion. In *Proceedings of ICALP*, volume 6756 of *LNCS*, pages 319–331. Springer, 2011.

[5] T. Brázdil, A. Kučera, and O. Stražovský. Deciding probabilistic bisimilarity over infinite-state probabilistic systems. *Acta Inf.*, 45(2):131–154, 2008.

[6] J. Canny. Some algebraic and geometric computations in PSPACE. In *STOC'88*, pages 460–467, 1988.

[7] D. Chistikov and R. Majumdar. Unary pushdown automata and straight-line programs. In *Proceedings of ICALP*, 2014. To appear.

[8] W. Czerwinski and S. Lasota. Fast equivalence-checking for normed context-free processes. In *FSTTCS 2010*, pages 260–271, 2010.

[9] J. Esparza, A. Kučera, and R. Mayr. Model checking probabilistic pushdown automata. In *LICS'04*, pages 12–21. IEEE, 2004.

[10] J. Esparza, A. Kučera, and R. Mayr. Quantitative analysis of probabilistic pushdown automata: Expectations and variances. In *LICS'05*, pages 117–126. IEEE, 2005.

[11] K. Etessami and M. Yannakakis. Recursive Markov chains, stochastic grammars, and monotone systems of nonlinear equations. *J. of the ACM*, 56(1):1–66, 2009.

[12] K. Etessami and M. Yannakakis. Model checking of recursive probabilistic systems. *ACM Trans. Comput. Logic*, 13(2):12:1–12:40, 2012.

[13] Vojtěch Forejt, Petr Jančar, Stefan Kiefer, and James Worrell. Bisimilarity of probabilistic pushdown automata. In Deepak D'Souza, Telikepalli Kavitha, and Jaikumar Radhakrishnan, editors, *IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2012, December 15-17, 2012, Hyderabad, India*, volume 18 of *LIPIcs*, pages 448–460. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2012.

[14] Y. Hirshfeld, M. Jerrum, and F. Moller. A polynomial algorithm for deciding bisimilarity of normed context-free processes. *Theor. Comput. Sci.*, 158(1&2):143–159, 1996.

[15] J. Honkala. Decision problems concerning algebraic series with noncommuting variables. In *Structures in Logic and Computer Science*, volume 1261 of *LNCS*, pages 281–290. Springer, 1997.

[16] J.E. Hopcroft, R. Motwani, and J.D. Ullman. *Introduction to Automata Theory, Languages, and Computation*. Addison Wesley, 3rd edition, 2007.

[17] S. Kiefer, A.S. Murawski, J. Ouaknine, and J. Worrell. On the complexity of equivalence and minimisation for Q-weighted automata. *Logical Methods in Computer Science*, 9(1:08):1–22, 2013.

[18] W. Kuich. On the multiplicity equivalence problem for context-free grammars. In *Results and Trends in Theoretical Computer Science*, volume 812 of *LNCS*, pages 232–250. Springer, 1994.

[19] W. Kuich and A. Salomaa. *Semirings, Automata, Languages*. Springer, 1986.

[20] J.M. Ortega. *Matrix theory: a second course*. Springer, 1987.

[21] D. Raz. Deciding multiplicity equivalence for certain context-free languages. In *Developments in Language Theory*, pages 18–29, 1993.

[22] J. Renegar. On the computational complexity and geometry of the first-order theory of the reals. Parts I–III. *J. of Symbolic Computation*, 13(3):255–352, 1992.

[23] D.J. Rosenkrantz. Matrix equations and normal forms for context-free grammars. *Journal of the ACM*, 14(3):501–507, 1967.

[24] G. Sénizergues. L(A)=L(B)? decidability results from complete formal systems. *Theor. Comput. Sci.*, 251(1-2):1–166, 2001.

[25] F.J. Urbanek. On Greibach normal form construction. *Theoretical Computer Science*, 40:315–317, 1985.