Information and Computation 237 (2014) 1–11

Contents lists available at ScienceDirect

Information and Computation

www.elsevier.com/locate/yinco

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

ARTICLE INFO

Article history:
Received 9 July 2013
Available online 28 April 2014

Keywords:
Pushdown systems
Language equivalence
Probabilistic systems

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.

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

* Corresponding author.

http://dx.doi.org/10.1016/j.ic.2014.04.003
0890-5401/

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 ($i = 0, 1, 2, \ldots$). 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 $support(d) := \{a \in A : d(a) > 0\}$. 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 $s' \in support(d)$. 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 $a_1 a_2 \ldots a_k \in \Sigma^*$, 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 $a_1 a_2 \ldots a_k$, 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 $\mathcal{P}^{\Delta}_{q\alpha}(w)$.

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.