Volume 1, Issue 5 December 2015 ISSN NO: 2454 - 7514

North Asian International Research Journal Consortium

North Asian International Research Journal

Of

Science, Engineering and Information Technology

Welcome to NAIRJC

ISSN NO: 2454 -7514

North Asian International Research Journal of Science, Engineering & Information Technology is a research journal, published monthly in English, Hindi, Urdu all research papers submitted to the journal will be double-blind peer reviewed referred by members of the editorial board. Readers will include investigator in Universities, Research Institutes Government and Industry with research interest in the general subjects

Editorial Board

Address: - Dr. Ashak Hussain Malik House No. 221 Gangoo, Pulwama, Jammu and Kashmir, India - 192301, Cell: 09086405302, 09906662570, Ph. No: 01933-212815,
Email: nairjc5@gmail.com, nairjc@nairjc.com, info@nairjc.com Website: www.nairjc.com

MULTI EQUIVELANCE PROBLEM OF PROBABILISTIC ALESHIN TYPE AUTOMATA

A.JEYANTHI[1]

[1]Faculty, Department of Mathematics, Anna University, Regional Office Madurai, Tamilnadu, India.

B.STALIN[2]

[2]Assistant Professor, Department of Mathematics, Anna University, Regional Office Madurai, Tamilnadu, India.

ABSTACT:

We study the multi equivalence problem for probabilistic Aleshin Type Automata (pAA) 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 pAA with one control state.In contrast, for the case of a one-letter input alphabet we show that pAA language E equivalence (and hence multiplicity equivalence of context-free grammars) is in PSPACE and at least as hard as the polynomial identity testing problem.

KEYWORDS: *Bisimulation probabilistic automata (BPA) ,probabilistic Aleshin automata , probabilistic labelled transition system (pLTS),*

1. INTRODUCTION

An input word determines a unique computation of a dAA, whereas the computation of a PAA on an input word can have many branches. In this paper we are concerned with probabilistic Aleshin type automata (pAA), where we only allow probabilistic branching. Here two pAA are language equivalent if they accept each word with the same probability. The decidability of the language equivalence problem for pAA is still open, even in the case with no ε-transitions, to which we restrict ourselves in this paper. The language theory of probabilistic Aleshin type 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 Aleshin type automata, probabilistic one-counter machines and probabilistic BPA (see, e.g., [5,10,11,13]). 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 Aleshin type automata (dAA) [24]; the problem is well-known to be undecidable for general (nondeterministic) PAA.It was shown recently in [17] that the language equivalence problem for probabilistic visibly pushdown automata is log space

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 pAA 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 pAA, and vice versa. A consequence of this reduction is that the equivalence problem for pAA is polynomially reducible to the equivalence problem for pAA 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 dAA 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, . . .). 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 pAA is at least as hard as the polynomial identity testing. Very recent work [7] considers (non)probabilistic dAA with a one-letter input alphabet, allowing for ε -transitions. They show, among other results, that the equivalence problem for such dAA 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 N, Q, R we denote the set of nonnegative integers, the set of rationals, and the set of real's, respectively. We denote the set of words over a finite alphabet Σ by Σ^*. We denote the empty word by ε and write $\Sigma^+ = \Sigma^* - \{\varepsilon\}$. By $|w|$ we denote the length of $w \in \Sigma^*$, so that $|\varepsilon| = 0$. For $k \in N$ 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 $d : A \to [0, 1] \cap Q$ 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 D(A). A Dirac distribution is one whose support is a singleton.

2.1 Construction of 4-state Aleshin type automaton S

The constructed Aleshin type automaton S, over the alphabet X= {0, 1} with the set of internal states Q ={a, b, c, d}. The state transition function φ and the output function ψ of S are defined as follows: δ (a,0)=d, δ (a,1)=b, δ (b,0)=b, δ (b,1)=c, δ(c,0)=d, δ(c,1)=d, δ(d,0)=a, δ(d,1)=a ; ψ(a,0)=1, ψ(a,1)=0, ψ(b,0)=1, ψ(b,1)=0, ψ(c,0)=0, ψ(c,1)=1, ψ(d,0)=0, ψ(d,1)=1.

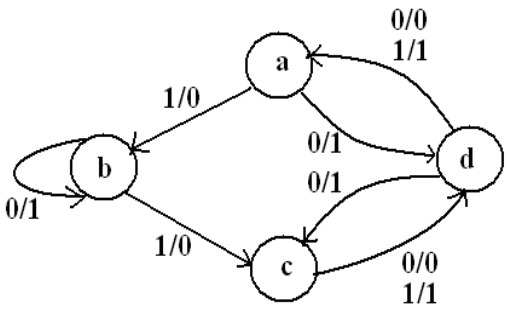

Fig. 1 Aleshin type automaton S

2.2 Probabilistic labeled transition systems

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

Let S = (S, Σ, →) be a pLTS. An execution on a word $a_1a_2 \ldots a_k \in \Sigma^*$, starting in a given state s_0 , is a finite sequence $s_0 \xrightarrow{ao,x_1} s_1 \xrightarrow{a_2,x_2} s_{2\ldots\ldots} \xrightarrow{a_k,x_k} s_k$. Given s_0 and $a_1a_2 \ldots a_k$, the probability of such an execution is $\prod_{i=1}^{k} x_i$.

2.3. Probabilistic Aleshin Type automata

A probabilistic pushdown automaton (pAA) is a tuple _ = (Q , Γ , Σ , →) where Q is a finite set of control states, Γ is a finite stack alphabet, Σ is a finite input alphabet, and → ⊆ Q × Γ × Σ × D(Q × $\Gamma^{\leq 2}$) is a finite set of rules. We require that for each (q, X, a) ∈ Q × Γ × Σ there be at most one distribution d such that (q, X ,a,d) ∈ →. We write qX $\xrightarrow{a}$ d to denote (q, X , a, d) ∈ →; 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 pAA Δ = (Q, Γ, Σ, →) is a pair (q, β) ∈ Q × Γ ∗ ; we often write qβ instead of (q, β). We write qX $\xrightarrow{a,x}$ rβ if qX $\xrightarrow{a}$ d where d(rβ) = 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 pAA Δ = (Q ,Γ ,Σ , →) generates a pLTS S (Δ) = (Q × Γ ∗ ,Σ , →) as follows. For each β ∈ Γ ∗ , a rule

qX $\xrightarrow{a}$ d of Δ induces a transition qXβ $\xrightarrow{a}$ d' in S (Δ), where d' ∈ D(Q × Γ∗) is defined by d'(pαβ) = d(pα) for all p ∈ Q and α ∈ Γ $^{\leq 2}$ (and thus d' is 0 elsewhere). We note that all configurations with the empty stack, written as pε or just as p , are terminating states in S (Δ). (Later we will assume that the empty-stack configurations are the only terminating states.)

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

A probabilistic basic process algebra (pAA)Δ is a pAA with only one control state. In this case we often write just α instead of qα for a configuration

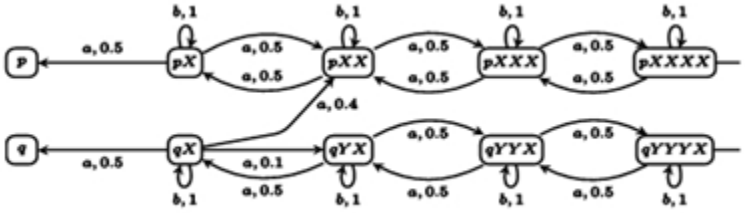

Fig.2A fragment of S(Δ)

2.4. The language equivalence problem

We study the language equivalence problem for pAA. The problem asks whether two configurations q_1 α_1 and q_2 α_2 of a given pAAΔ accept each input word with the same probability, i.e., whether the functions P $\Delta q_1\alpha_1$ (·) and P $\Delta q_2\alpha_2$ (·) are the same. If yes, we also say that q1 α1 and q2 α2 are equivalent in Δ.

Example 1. Consider the pAAΔ = ({p,q}, {X , Y }, {a,b}, _→) with the following rules:

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

The restriction of Δ to the control state p yields a Probability Of bisimilarity automata pBPA. A fragment of the pLTS S (Δ) is shown in Fig. 2. The configurations pX X and qY X are equivalent in Δ, since for every word w we have P $\Delta_{pX\,X}$ (w) = P Δ_{qYX} (w); this can be derived by observing that for all words w and i ≥ 1 the probability of being in p X^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 pAA Δ= (Q ,Γ ,Σ , →) is non-blocking if for each (q, X ,a) ∈ Q × Γ × Σ there is

(precisely) one distribution d such that $qX \xrightarrow{a} d$; hence each action is disabled only in the empty-stack configurations in S (Δ). We assume that our pAA are non-blocking. Given an arbitrary pAA $\Delta = (Q , \Gamma , \Sigma , \rightarrow)$ we obtain an equivalent non-blocking pAA Δ′ as follows: we add a fresh stack symbol ⊥ and for every (q, X ,a) where there is no d such that (q, X ,a,d) ∈ → (which includes the case X = ⊥) we add the rule (q, X ,a,d) where d(q⊥) = 1. Hence $P\,\Delta'_{qa}(w) = 0$ if a contains ⊥, and $P\,\Delta_{qa}(w) = P\,\Delta'_{qa}(w)$ 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.

2.5.Grammars 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 $A \rightarrow \alpha$ where $\alpha \neq \varepsilon$, i.e., we assume that grammars are ε-free. The relation ⇒ on $(V \cup \Sigma)*$, capturing a derivation step, is defined as follows: if $A \rightarrow \alpha$ 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 $i \in \{0, 1, \ldots ,k - 1\}$.

2.6. Multiplicity problem

For a grammar $G = (V ,\Sigma , R , S)$ and a word $w \in \Sigma *$ we define the multiplicity $m_G (w) \in N \cup \{\infty\}$ 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-ultiplicity grammars G_1 , G_2 are said to be multiplicity equivalent if $m_{G1}(w) = m_{G2}(w)$ for all $w \in \Sigma^*$. If a grammar has a rule $S \rightarrow S$ 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.

2.8. 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 pAA;
2. language equivalence for pBPA;
3. multiplicity equivalence for (ε -free) finite-multiplicity grammars.

Theorem 2. The language equivalence problem for pAA with a one-letter input alphabet, the language equivalence problem for pAA 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).

2.9. 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 $\{x_i : i \in N\}$. 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^{2n} , 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 pAA WITH ONE INPUT LETTER

In this section we consider unary pAA _ = (Q , Γ , {a}, →), i.e., those whose input alphabet is a singleton {a}. In unary pAA the next configuration is probabilistically determined solely by the current configuration. Here we elide the letter a in transitions, writing $qX \xrightarrow{p} r\alpha$ instead of $qX \xrightarrow{a,p} r\alpha$, and $q\beta \xrightarrow{p} r\gamma$ instead of $q\beta \xrightarrow{a,p} r\gamma$.

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 pAA is in PSPACE.

We show this by a polynomial reduction to the decision problem for ExTh(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 pAA is not PSPACE-hard unless ExTh(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(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 pAA by using the direction 2. $\Rightarrow$ 1.

3.1.1. Distribution of termination time of runs

Let us assume a fixed unary (non-blocking) pAAΔ = (Q , Γ , {a}, $\rightarrow$). We note that our model of unary pAA is essentially equivalent to a model which is also called "pAA" in the literature, see e.g., [4] and the references therein. As there is only one action in our unary pAA Δ, the pLTS S (Δ) can be viewed as an infinite-state Markov chain. Let a run of Δ be an execution in S (Δ) 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 qα , by Run(qα) we denote the set of runs starting in qα . A probability measure P can be defined over Run(qα) in the standard way, see e.g., [4] for the formal details.

To each configuration qα we associate the random variable $T_{q\alpha}$: Run(qα) $\rightarrow$ N $\cup$ {∞} that maps each run to the number of its steps, called the termination time of the run. For each i $\in$ N we have that P($T_{q\alpha}$ = i), i.e., the probability that a run from qα terminates in i steps, is equal to P$\Delta_{q\alpha}$ (ai). From this point of view, language equivalence means equality between the (sub-)distributions of $T_{q1\,\alpha1}$ and $T_{q2\,\alpha2}$. We note that some bounds on 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 Δ = ({q}, Γ , {a}, $\rightarrow$), i.e., we assume P(Tα < ∞) = 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 $\in$ N such that P($T_{X = i}$)$\neq$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 N we define $g_T : [0,1] \rightarrow [0, 1]$ by

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

Where E denotes the expectation with respect to 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 $P(T = i) = P(T' = i)$. 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

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

We further assume an almost surely terminating pBPAΔ $= (\{q\}, \Gamma , \{a\}, \rightarrow)$. For $\alpha \in \Gamma *$ we often write g_α instead of $g_{T\alpha}$.

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

$$\sum_{i=0}^{\infty} P(T_X + T_Y = i).z^i \sum_{i=0}^{\infty}\sum_{j=0}^{\infty} P(T_X = j).P(T_Y = i - j)z^j.z^{i-j}$$

$$= \sum_{i=0}^{\infty} P(T_X = i).z^i \sum_{i=0}^{\infty} P(T_Y = i).z^i$$

We illustrate our equation systems by an example first.

Example2. Letus consider consider a unary pBPA given by

$$X \xrightarrow{0.3} XY,\ X \xrightarrow{0.7} \varepsilon \text{ and } Y \xrightarrow{0.6} X\ ,\ Y \xrightarrow{0.4} \varepsilon$$

We can easily check that

$g_{X(z)} = E(z^{TX}) = 0.3 \cdot E(z^{TX} |\text{rule } X \xrightarrow{0.3} XY \text{ is taken in the first step}$
$+ 0.7 \cdot E(z^{TX}| \text{ rule } X \xrightarrow{0.7} \varepsilon \text{ is taken in the first step})$
$= 0.3 \cdot E(z^{1+TX+TY}) + 0.7 \cdot E(z^1) \quad = z \cdot (0.3 \cdot g_X(z) \cdot g_Y(z) + 0.7).$

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 $z \cdot f$
where $f(x_1 , x_2) = (0.3x_1 x_2 + 0.7, 0.6x_1 + 0.4)$.

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

$$f_X(x) := \sum_{X \xrightarrow{p} YZ} pXyXz + \sum_{X \xrightarrow{p} YZ} pXy + \sum_{X \xrightarrow{p} \varepsilon} p \quad \text{for } x \in [0, 1]\Gamma . \qquad (1)$$

Reasoning as in Example 2, we note that the vector $g(z) := (g_X(z))_{X \in \Gamma}$ satisfies $g(z) = z \cdot f(g(z))$ for each $z \in [0, 1]$, i.e., $g(z)$ is a fixed point of $z \cdot f$. 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 $F : R^k \rightarrow R^k$, i.e., a k-tuple $(F_1, F_2, \ldots, F_k)$ with $F_i : R^k \rightarrow R$. Assume moreover that the partial derivatives of each F_i exist throughout R^k. For $x = (x_1, x_2, \ldots, x_k)$, we denote by $F'(x)$ the Jacobian matrix of $F(x)$, i.e., the

$(k \times k)$-matrix with $F'_{ij}(x) := \frac{\partial}{\partial x_j} F_i(x)$.

CONCLUSION

We have proved the problems are interreducible in polynomial time, without changing the input alphabet or terminal alphabet, respectively same or equal in language equivalence for pAA; language equivalence for pBPA, multiplicity equivalence for (ε-free) finite-multiplicity grammars. The two identical equation systems for generating functions, with unique solutions are the same.

REFERENCES

[1] S.P. Abney, D.A. McAllester, F. Pereira, Relating probabilistic grammars and automata, in: ACL, ACL, 1999, pp. 542-549.

[2] E.E.llender, P. Bürgisser, J. Kjeldgaard-Pedersen, P. Bro Miltersen, On the complexity of numerical analysis, SIAM J. Comput. 38 (5) (2009) 1987-2 006. [3] A0. Berman, R.J. Plemmons, Nonnegative Matrices in the Mathematical Sciences, SIAM, 1994.

[4] T6. Brázdil, S. Kiefer, A. Kuera, I.H. Vaˇeková, Runtime analysis of probabilistic programs with unbounded recursion, in: Proceedings of ICALP, in: Lect. Notes Comput. Sci., vol. 6756, Springer, 2011, pp. 319-331.

[5] T.Brázdil, A. Ku ˇera, O. Stražovský, Deciding probabilistic bisimilarity over infinite-state probabilistic systems, Acta Inform. 45 (2) (2008) 131-154.

[6] J.4Canny, Some algebraic and geometric computations in PSPACE, in: STOC'88, 1988, pp. 460-467.

[7] D.Chistikov, R. Majumdar, Unary pushdown automata and straight-line programs, in: Proceedings of ICALP, 2014, in press.

[8] W. Czerwinski, S. Lasota, Fast equivalence-checking for normed context-free processes, in: FSTTCS 2010, 2010, pp. 260-271.

[9] J.4Esparza, A. Kuˇera, R. Mayr, Model checking probabilistic pushdown automata, in: LICS'04, IEEE, 2004, pp. 12-21.

[10] J.Esparza, A. Kuˇera, R. Mayr, Quantitative analysis of probabilistic pushdown automata: expectations and variances, in: LICS'05, IEEE,2005,pp. 117-126.
[11] K.tessami, M. Yannakakis, Recursive Markov chains, stochastic grammars, and monotone systems of nonlinear equations, J. ACM 56 (1) (2009) 1-66. [12] K.tessami, M. Yannakakis, Model checking of recursive probabilistic systems, ACM Trans. Comput. Log. 12 (2012) 12:1-12:40.
[13] Vojt ˇch Forejt, Petr Jan ˇar, Stefan Kiefer, James Worrell, Bisimilarity of probabilistic pushdown automata, in: Deepak D'Souza, Telikepalli Kavitha, Jaikumar Radhakrishnan (Eds.), IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2012, December 15-17, 2012, Hyderabad, India, in: LIPIcs, vol. 18, Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2012, pp. 448-460.
[14] Y Hirshfeld, M. Jerrum, F. Moller, A polynomial algorithm for deciding bisimilarity of normed context-free processes, Theor. Comput. Sci. 158 (1&2) (1996)143-159.
[15] J.Honkala, Decision problems concerning algebraic series with noncommuting variables, in: Structures in Logic and Computer Science, in: Lect. Notes Comput. Sci., vol. 1261, Springer, 1997, pp. 281-290.
[16] J.E.opcroft, R. Motwani, J.D. Ullman, Introduction to Automata Theory, Languages, and Computation, 3rd edition, Addison Wesley, 2007.
[17] S. iefer, A.S. Murawski, J. Ouaknine, J. Worrell, On the complexity of equivalence and minimisation for Q-weighted automata, Log. Methods Comput.Sci. 1 (2013) 08.
[18] WKuich, On the multiplicity equivalence problem for context-free grammars, in: Results and Trends in Theoretical Computer Science, in: Lect. Notes Comput. Sci., vol. 812, Springer, 1994, pp. 232-250.
[19] WKuich, A. Salomaa, Semirings, Automata, Languages, Springer, 1986.
[20] J.M. ega, Matrix Theory: A Second Course, Springer, 1987.
[21] D.az, Deciding multiplicity equivalence for certain context-free languages, in: Developments in Language Theory, 1993, pp. 18-29.
[22] J.Renegar, On the computational complexity and geometry of the first-order theory of the reals. Parts I-III, J. Symb. Comput. 13 (23) (1992) 255-352.
[23] D.J.senkrantz, Matrix equations and normal forms for context-free grammars, J. ACM 14 (3) (1967) 501-507.
[24] G.énizergues, L(A)=L(B)? decidability results from complete formal systems, Theor. Comput. Sci. 251 (1-2) (2001) 1-166. [25] FJ. Urbanek, On Greibach normal form construction, Theor. Comput. Sci. 40 (1985) 315-317.

Publish Research Article

Dear Sir/Mam,

We invite unpublished Research Paper,Summary of Research Project,Theses,Books and Book Review for publication.

Address:- Dr. Ashak Hussain Malik House No-221, Gangoo Pulwama - 192301 Jammu & Kashmir, India
Cell: 09086405302, 09906662570,
Ph No: 01933212815
Email:- nairjc5@gmail.com, nairjc@nairjc.com , info@nairjc.com
Website: www.nairjc.com