FI MU

Faculty of Informatics
Masaryk University Brno

On the Decidability of Temporal Properties of Probabilistic Pushdown Automata

by

Tomáš Brázdil
Antonín Kučera
Oldřich Stražovský

FI MU Report Series **FIMU-RS-2005-01**

Copyright © 2005, FI MU **February 2005**

**Copyright © 2005, Faculty of Informatics, Masaryk University.
All rights reserved.**

**Reproduction of all or part of this work
is permitted for educational or research use
on condition that this copyright notice is
included in any copy.**

Publications in the FI MU Report Series are in general accessible via WWW:

`http://www.fi.muni.cz/veda/reports/`

Further information can obtained by contacting:

**Faculty of Informatics
Masaryk University
Botanická 68a
602 00 Brno
Czech Republic**

On the Decidability of Temporal Properties of Probabilistic Pushdown Automata

Tomáš Brázdil* Antonín Kučera† Oldřich Stražovský

Abstract

We consider qualitative and quantitative model-checking problems for probabilistic pushdown automata (pPDA) and various temporal logics. We prove that the qualitative and quantitative model-checking problem for ω-regular properties and pPDA is in **2-EXPSPACE** and **3-EXPTIME**, respectively. We also prove that model-checking the qualitative fragment of the logic PECTL* for pPDA is in **2-EXPSPACE**, and model-checking the qualitative fragment of PCTL for pPDA is in **EXPSPACE**. Furthermore, model-checking the qualitative fragment of PCTL is shown to be **EXPTIME**-hard even for stateless pPDA. Finally, we show that PCTL model-checking is undecidable for pPDA, and PCTL$^+$ model-checking is undecidable even for stateless pPDA.

1 Introduction

In this paper we concentrate on a subclass of discrete probabilistic systems (see, e.g., [Kwi03]) that correspond to probabilistic sequential programs with recursive procedure calls. Such programs can conveniently be modeled by probabilistic pushdown automata (pPDA), where the stack symbols correspond to procedures and global data is stored in the finite control. This model is equivalent to probabilistic recursive state machines, or recursive Markov chains (see, e.g., [AEY01, EY05, EY]). An important subclass of pPDA are stateless pPDA, denoted pBPA[1]. In the non-probabilistic setting, BPA are often easier to analyze than general PDA (i.e., the corresponding algorithms are more efficient),

*Supported by the Grant Agency of the Czech Republic, grant No. 201/03/1161.

†Supported by the Alexander von Humboldt Foundation and by the research centre "Institute for Theoretical Computer Science (ITI)", project No. 1M0021620808.

[1]This notation is borrowed from process algebra; stateless PDA correspond (in a well-defined sense) to processes of the so-called Basic Process Algebra.

but they still retain a reasonable expressive power which is sufficient, e.g., for modelling some problems of interprocedural dataflow analysis [EK99]. There is a close relationship between pBPA and stochastic context-free grammars. In fact, pBPA *are* stochastic context-free grammars, but they are seen from a different perspective in the setting of our paper. We consider the model-checking problem for pPDA/pBPA systems and properties expressible in probabilistic extensions of various temporal logics.

The State of the Art. Methods for automatic verification of probabilistic systems have so far been examined mainly for finite-state probabilistic systems. Model-checking algorithms for various (probabilistic) temporal logics like LTL, PCTL, PCTL*, probabilistic μ-calculus, etc., have been presented in [LS82, HS84, Var85, HJ94, ASB$^+$95, CY95, HK97, CSS03]. As for infinite-state systems, most works so far considered probabilistic lossy channel systems [IN97] which model asynchronous communication through unreliable channels [BE99, ABIJ00, AR03, BS03, Rab03]. The problem of deciding probabilistic bisimilarity over various classes of infinite-state probabilistic systems has recently been considered in [BKS04]. Model-checking problems for pPDA and pBPA processes have been studied in [EKM04]. In [EKM04], it has been shown that the qualitative/quantitative random walk problem for pPDA is in **EXPTIME**, that the qualitative fragment of the logic PCTL is decidable for pPDA (but no upper complexity bound was given), and that the qualitative/quantitative model-checking problem for pPDA and a subclass of ω-regular properties definable by deterministic Büchi automata is also decidable. The reachability problem for pPDA and pBPA processes is studied in greater depth in [EY05], where it is shown that the qualitative reachability problem for pBPA is solvable in polynomial time, and a fast-converging algorithm for quantitative pPDA reachability is given.

Our Contribution. In this paper we continue the study initiated in [EKM04]. We still concentrate mainly on clarifying the decidability/undecidability border for model-checking problems, but we also pay attention to complexity issues. Basic definitions together with some useful existing results are recalled in Section 2. As a warm-up, in Section 3 we show that both qualitative and quantitative model-checking problem for ω-regular properties and pPDA is decidable. More precisely, if ω-regular properties are encoded by Büchi automata, then the qualitative variant of the problem is in **2-EXPSPACE**, and the quantitative one is in **3-EXPTIME**. The proof is obtained by extending and modifying the construction for deterministic Büchi automata given in [EKM04] so that it works for Muller automata. Note that the considered problems are

known to be **PSPACE**-hard even for finite-state systems [Var85]. The core of the paper is Section 4. First we prove that model-checking general PCTL is undecidable for pPDA, and model-checking PCTL$^+$ is undecidable even for pBPA. Since the structure of formulae which are constructed in our proofs is relatively simple, our undecidability results hold even for fragments of these logics. From a certain point of view, these results are tight (see Section 4). Note that in the non-probabilistic case, the model-checking problems for logics like CTL, CTL*, or even the modal μ-calculus, are decidable for PDA. Our undecidability proofs are based on a careful arrangement of transition probabilities in the constructed pPDA so that various nontrivial properties can be encoded by specifying probabilities of certain events (which are expressible in PCTL or PCTL$^+$). We believe that these tricks might be applicable to other problems and possibly also to other models. In the light of these undecidability results, it is sensible to ask if the model-checking problem is decidable at least for some natural fragments of probabilistic branching-time logics. We show that model-checking the *qualitative fragment* of the logic PECTL* is decidable for pPDA, and we give the **2-EXPSPACE** upper bound. For the qualitative fragment of PCTL we give the **EXPSPACE** upper bound. We also show that model-checking the qualitative fragment of PCTL is **EXPTIME**-hard even for pBPA processes. Our proof is a simple modification of the one given in [Wal00] which shows **EXPTIME**-hardness of the model-checking problem for (non-probabilistic) CTL and PDA.

2 Preliminaries

For every alphabet Σ, the symbols Σ^* and Σ^ω denote the sets of all finite and infinite words over the alphabet Σ, respectively. The length of a given $w \in \Sigma^* \cup \Sigma^\omega$ is denoted $|w|$ (if $w \in \Sigma^\omega$ then we put $|w| = \omega$). For every $w \in \Sigma^* \cup \Sigma^\omega$ and every $0 \leq i < |w|$, the symbols $w(i)$ and w_i denote the i+1-th letter of w and the suffix of w which starts with $w(i)$, respectively. By writing $w(i)$ or w_i we implicitly impose the condition that the object exists.

Definition 2.1. *A* Büchi *automaton is a tuple* $\mathcal{B} = (\Sigma, B, \rho, b_I, Acc)$, *where* Σ *is a finite* alphabet, B *is a finite set of* states, $\rho \subseteq B \times \Sigma \times B$ *is a* transition relation *(we write* $b \xrightarrow{a} b'$ *instead of* $(b, a, b') \in \rho$*),* b_I *is the* initial state, *and* $Acc \subseteq B$ *is a set of* accepting states.

A word $w \in \Sigma^\omega$ is *accepted* by $\mathcal{B}$ if there is a run of $\mathcal{B}$ on w which visits some accepting state infinitely often. The set of all $w \in \Sigma^\omega$ which are accepted by $\mathcal{B}$ is denoted $L(\mathcal{B})$.

Definition 2.2. *A* probabilistic transition system *is a triple* $\mathcal{T} = (S, \to, Prob)$ *where* S *is a finite or countably infinite set of* states, $\to \subseteq S \times S$ *is a* transition relation, *and Prob is a function which to each transition* $s \to t$ *of* $\mathcal{T}$ *assigns its probability* $Prob(s \to t) \in (0, 1]$ *so that for every* $s \in S$ *we have that* $\sum_{s \to t} Prob(s \to t) \in \{0, 1\}$. *(The sum above can be* 0 *if* s *does not have any outgoing transitions.)*

In the rest of this paper we write $s \xrightarrow{x} t$ instead of $Prob(s \to t) = x$. A *path* in $\mathcal{T}$ is a word $w \in S^* \cup S^\omega$ such that $w(i{-}1) \to w(i)$ for every $1 \leq i < |w|$. A *run* is a maximal path, i.e., a path which cannot be prolonged. The sets of all finite paths, all runs, and all infinite runs of $\mathcal{T}$ are denoted *FPath*, *Run*, and *IRun*, respectively[2]. Similarly, the sets of all finite paths, runs, and infinite runs that start in a given $s \in S$ are denoted $FPath(s)$, $Run(s)$, and $IRun(s)$, respectively.

Each $w \in FPath$ determines a *basic cylinder* $Run(w)$ which consists of all runs that start with w. To every $s \in S$ we associate the probabilistic space $(Run(s), \mathcal{F}, \mathcal{P})$ where $\mathcal{F}$ is the σ-field generated by all basic cylinders $Run(w)$ where w starts with s, and $\mathcal{P} : \mathcal{F} \to [0, 1]$ is the unique probability function such that $\mathcal{P}(Run(w)) = \prod_{i=1}^{|w|-1} x_i$ where $w(i{-}1) \xrightarrow{x_i} w(i)$ for every $1 \leq i < |w|$ (if $|w| = 1$, we put $\mathcal{P}(Run(w)) = 1$).

The logics PCTL, PCTL$^+$, PCTL*, PECTL*, and their qualitative fragments.

Let $Ap = \{a, b, c, \dots\}$ be a countably infinite set of *atomic propositions*. The syntax of PCTL* *state* and *path* formulae is given by the following abstract syntax equations (for simplicity, we omit the bounded 'until' operator from the syntax of path formulae).

$$\begin{aligned} \Phi &::= \mathtt{tt} \mid a \mid \neg\Phi \mid \Phi_1 \wedge \Phi_2 \mid \mathcal{P}^{\sim\rho}\varphi \\ \varphi &::= \Phi \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid \mathcal{X}\varphi \mid \varphi_1 \,\mathcal{U}\, \varphi_2 \end{aligned}$$

Here a ranges over Ap, $\rho \in [0, 1]$, and $\sim \in \{\leq, <, \geq, >\}$. The logic PCTL is a fragment of PCTL* where state formulae are defined as for PCTL* and path formulae are given by the equation $\varphi ::= \mathcal{X}\Phi \mid \Phi_1 \,\mathcal{U}\, \Phi_2$. The logic PCTL$^+$ is a fragment of PCTL* where the $\mathcal{X}$ and $\mathcal{U}$ operators in path formulae can be combined using Boolean connectives, but they cannot be nested. Finally, the logic PECTL* is an extension of PCTL* where only state formulae are introduced and have the following syntax:

$$\Phi ::= \mathtt{tt} \mid a \mid \neg\Phi \mid \Phi_1 \wedge \Phi_2 \mid \mathcal{P}^{\sim\rho}\mathcal{B}$$

[2] In this paper, $\mathcal{T}$ is always clear from the context.

Here $\mathcal{B}$ is a Büchi automaton over an alphabet $2^{\{\Phi_1,\cdots,\Phi_n\}}$, where each Φ_i is a PECTL* formula.

Let $\mathcal{T} = (S, \to, \mathit{Prob})$ be a probabilistic transition system, and let $\nu : \mathit{Ap} \to 2^S$ be a *valuation*. The semantics of PCTL* is defined below. State formulae are interpreted over S, and path formulae are interpreted over *IRun*. (Alternatively, path formulae could also be interpreted over *Run*. This would not lead to any problems, and our model-checking algorithms would still work after some minor modifications. We stick to infinite runs mainly for the sake of simplicity.)

$$
\begin{array}{lll}
s \models^\nu \mathtt{tt} & & \\
s \models^\nu a & \text{iff} & s \in \nu(a) \\
s \models^\nu \neg\Phi & \text{iff} & s \not\models^\nu \Phi \\
s \models^\nu \Phi_1 \wedge \Phi_2 & \text{iff} & s \models^\nu \Phi_1 \text{ and } s \models^\nu \Phi_2 \\
s \models^\nu \mathcal{P}^{\sim\rho}\varphi & \text{iff} & \mathcal{P}(\{w \in \mathit{IRun}(s) \mid w \models^\nu \varphi\}) \sim \rho
\end{array}
\qquad
\begin{array}{lll}
w \models^\nu \Phi & \text{iff} & w(0) \models^\nu \Phi \\
w \models^\nu \neg\varphi & \text{iff} & w \not\models^\nu \varphi \\
w \models^\nu \varphi_1 \wedge \varphi_2 & \text{iff} & w \models^\nu \varphi_1 \text{ and } w \models^\nu \varphi_2 \\
w \models^\nu \mathcal{X}\varphi & \text{iff} & w_1 \models^\nu \varphi \\
w \models^\nu \varphi_1 \,\mathcal{U}\, \varphi_2 & \text{iff} & \exists j \geq 0 : w_j \models^\nu \varphi_2 \text{ and} \\
 & & w_i \models^\nu \varphi_1 \text{ for all } 0 \leq i < j
\end{array}
$$

For PCTL, the semantics of path formulae is redefined to

$$
\begin{array}{lll}
w \models^\nu \mathcal{X}\Phi & \text{iff} & w(1) \models^\nu \Phi \\
w \models^\nu \Phi_1 \,\mathcal{U}\, \Phi_2 & \text{iff} & \exists j \geq 0 : w(j) \models^\nu \Phi_2 \text{ and } w(i) \models^\nu \Phi_1 \text{ for all } 0 \leq i < j
\end{array}
$$

The semantics of a PECTL* formula $\mathcal{P}^{\sim\rho}\mathcal{B}$, where $\mathcal{B}$ is a Büchi automaton over an alphabet $2^{\{\Phi_1,\cdots,\Phi_n\}}$, is defined as follows. First, we can assume that the semantics of the PECTL* formulae $\Phi_1, \cdots, \Phi_n$ has already been defined. This means that for each $w \in \mathit{IRun}$ we can define an infinite word $w_{\mathcal{B}}$ over the alphabet $2^{\{\Phi_1,\cdots,\Phi_n\}}$ by $w_{\mathcal{B}}(i) = \{\Phi \in \{\Phi_1, \cdots, \Phi_n\} \mid w(i) \models^\nu \Phi\}$. For every state s, let $\mathit{Run}(s, \mathcal{B}) = \{w \in \mathit{IRun}(s) \mid w_{\mathcal{B}} \in L(\mathcal{B})\}$. We stipulate that $s \models^\nu \mathcal{P}^{\sim\rho}\mathcal{B}$ iff $\mathcal{P}(\mathit{Run}(s, \mathcal{B})) \sim \rho$.

The *qualitative fragments* of PCTL, PCTL*, and PECTL*, denoted qPCTL, qPCTL*, and qPECTL*, resp., are obtained by restricting the allowed operator/number combinations in $\mathcal{P}^{\sim\rho}\varphi$ and $\mathcal{P}^{\sim\rho}\mathcal{B}$ subformulae to '≤ 0' and '≥ 1', which can also be written as '$= 0$' and '$= 1$', resp. (Observe that '< 1', '> 0' are definable from '≤ 0', '≥ 1', and negation.)

Probabilistic PDA.

A *probabilistic pushdown automaton (pPDA)* is a tuple $\Delta = (Q, \Gamma, \delta, \mathit{Prob})$ where Q is a finite set of *control states*, Γ is a finite *stack alphabet*, $\delta \subseteq Q \times \Gamma \times Q \times \Gamma^*$ is a finite *transition relation* (we write $pX \to q\alpha$ instead of $(p, X, q, \alpha) \in \delta$), and *Prob* is a function which to

each transition $pX \to q\alpha$ assigns its probability $Prob(pX \to q\alpha) \in (0,1]$ so that for all $p \in Q$ and $X \in \Gamma$ we have that $\sum_{pX \to q\alpha} Prob(pX \to q\alpha) \in \{0,1\}$.

A *pBPA* is a pPDA with just one control state. Formally, a pBPA is understood as a triple $\Delta = (\Gamma, \delta, Prob)$ where $\delta \subseteq \Gamma \times \Gamma^*$.

In the rest of this paper we adopt a more intuitive notation, writing $pX \xrightarrow{x} q\alpha$ instead of $Prob(pX \to q\alpha) = x$. The set $Q \times \Gamma^*$ of all configurations of Δ is denoted by $\mathcal{C}(\Delta)$. We also assume (w.l.o.g.) that if $pX \to q\alpha \in \delta$, then $|\alpha| \leq 2$. Given a configuration $pX\alpha$ of Δ, we call pX the *head* and α the *tail* of $pX\alpha$. To Δ we associate the probabilistic transition system $\mathcal{T}_\Delta$ where $\mathcal{C}(\Delta)$ is the set of states and the probabilistic transition relation is determined by $pX\beta \xrightarrow{x} q\alpha\beta$ iff $pX \xrightarrow{x} q\alpha$.

The model checking problem for pPDA configurations and any nontrivial class of properties is clearly undecidable for general valuations. Therefore, we restrict ourselves to *simple* valuations where the (in)validity of atomic propositions depends just on the current control state and the current symbol on top of the stack. Alternatively, we could consider *regular* valuations where the set of all configurations that satisfy a given atomic proposition is encoded by a finite-state automaton. However, regular valuations can be "encoded" into simple valuations by simulating the finite-state automata in the stack (see, e.g., [EKS03]), and therefore they do not bring any extra expressive power.

Definition 2.3. *A valuation ν is* simple *if there is a function f_ν which assigns to every atomic proposition a subset of $Q \times \Gamma$ such that for every configuration $p\alpha$ and every $a \in Ap$ we have that $p\alpha \models^\nu a$ iff $\alpha = X\alpha'$ and $pX \in f_\nu(a)$.*

Random Walks on pPDA Graphs.

Let $\mathcal{T} = (S, \to, Prob)$ be a probabilistic transition system. For all $s \in S$, $\mathcal{C}_1, \mathcal{C}_2 \subseteq S$, let $Run(s, \mathcal{C}_1 \mathcal{U} \mathcal{C}_2) = \{w \in Run(s) \mid \exists j \geq 0 : w(j) \in \mathcal{C}_2 \text{ and } w(i) \in \mathcal{C}_1 \text{ for all } 0 \leq i < j\}$. An instance of the *random walk problem* is a tuple $(s, \mathcal{C}_1, \mathcal{C}_2, \sim, \rho)$, where $s \in S$, $\mathcal{C}_1, \mathcal{C}_2 \subseteq S$, $\sim \in \{\leq, <, \geq, >, =\}$, and $\rho \in [0,1]$. The question is if $\mathcal{P}(Run(s, \mathcal{C}_1 \mathcal{U} \mathcal{C}_2)) \sim \rho$. In [EKM04], it was shown that the random walk problem for pPDA processes and simple sets of configurations is decidable (a simple set is a set of the form $\bigcup_{pX \in \mathcal{H}} \{pX\alpha \mid \alpha \in \Gamma^*\}$ where $\mathcal{H}$ is a subset of $Q \times \Gamma$). More precisely, it was shown that for a given tuple $(pX, \mathcal{C}_1, \mathcal{C}_2, \sim, \rho)$, where $\mathcal{C}_1, \mathcal{C}_2$ are simple sets of configurations of a given pPDA system Δ, there is an efficiently constructible system of recursive quadratic equations such that the probability $\mathcal{P}(Run(pX, \mathcal{C}_1 \mathcal{U} \mathcal{C}_2))$ is the first component in the tuple of non-negative real values which

form the least solution of the system. Thus, the relation $\mathcal{P}(Run(pX, \mathcal{C}_1 \mathcal{U} \mathcal{C}_2)) \sim \rho$ can effectively be expressed in $(\mathbb{R}, +, *, \leq)$ by constructing a formula Φ saying that a given vector $\vec{x}$ is the least solution of the system and $\vec{x}(1) \sim \rho$. Since the quantifier alternation depth in the constructed formula is fixed, it was concluded in [EKM04] that the random walk problem for pPDA and simple sets of configurations is in **EXPTIME** by applying the result of [Gri88]. Later, it was observed in [EY05] that the existential fragment of $(\mathbb{R}, +, *, \leq)$ is sufficient to decide the quantitative reachability problem for pPDA. This observation applies also to the random walk problem. Actually, it follows easily from the results of [EKM04] just by observing that the existential (or universal) fragment of $(\mathbb{R}, +, *, \leq)$ is sufficient to decide whether $\mathcal{P}(Run(pX, \mathcal{C}_1 \mathcal{U} \mathcal{C}_2)) \sim \rho$ when $\sim \in \{<, \leq\}$ (or $\sim \in \{>, \geq\}$, resp.). Since the existential and universal fragments of $(\mathbb{R}, +, *, \leq)$ are decidable in polynomial space [Can88], we obtain the following result which is used in our complexity estimations:

Lemma 2.4. *The random walk problem for pPDA processes and simple sets of configurations is in **PSPACE**.*

3 Model-Checking ω-regular Properties

In this section we show that the qualitative and quantitative model-checking problems for pPDA and ω-regular properties represented by Büchi automata are in **2-EXPSPACE** and **3-EXPTIME**, respectively. For both of these problems there is a **PSPACE** lower complexity bound due to [Var85]. Our proof is a generalization of the construction for *deterministic* Büchi automata presented in [EKM04]. We show that this construction can be extended to (deterministic) Muller automata, which have the same expressive power as general Büchi automata.

Definition 3.1. *A* Muller automaton *is a tuple $\mathcal{M} = (\Sigma, M, \rho, m_I, \mathcal{F})$, where Σ is a finite* alphabet, *M is a finite set of* states, *$\rho\colon M \times \Sigma \to M$ is a (total)* transition function *(we write $m \xrightarrow{a} m'$ instead of $\rho(m, a) = m'$), m_I is the* initial state, *and $\mathcal{F} \subseteq 2^M$ is a set of* accepting sets.

For every infinite run ν of $\mathcal{M}$, let $\inf(\nu)$ be the set of all states which appear in ν infinitely often. A word $w \in \Sigma^\omega$ is accepted *by $\mathcal{M}$ if $\inf(\nu) \in \mathcal{F}$, where ν is the (unique) run of $\mathcal{M}$ on w.*

For the rest of this section, we fix a pPDA $\Delta = (Q, \Gamma, \delta, Prob)$. We consider specifications given by Muller automata $\mathcal{M}$ having $Q \times \Gamma$ as their alphabet. Each infinite run

w of Δ determines a unique word $v \in (Q \times \Gamma)^\omega$, where $v(i)$ is the head of $w(i)$ for every $i \in \mathbb{N}_0$. A run w of Δ is *accepted* by $\mathcal{M}$ if its associated word v is accepted by $\mathcal{M}$. For a given configuration pX, let $Run(pX, \mathcal{M})$ be the set of all runs of $IRun(pX)$ that are accepted by $\mathcal{M}$. Our aim is to show that the problem if $\mathcal{P}(Run(pX, \mathcal{M})) \sim \rho$ for given Δ, pX, $\mathcal{M}$, $\sim \in \{\leq, <, \geq, >\}$, and $\rho \in [0, 1]$, is in **2-EXPTIME**. In the qualitative case, we derive the **EXPSPACE** upper bound.

Let us note that our results also apply to LTL formulae where atomic propositions are interpreted by regular valuations (using the results of [EKS03], regular valuations can effectively be replaced with valuations which take into account just heads of configurations).

Let $\mathcal{M} = (\mathcal{H}(\Delta), M, \rho, m_I, \mathcal{F})$ be a Muller automaton. The *product* of Δ and $\mathcal{M}$ is a pPDA $\Delta_{\mathcal{M}} = (Q \times M, \Gamma, \delta', Prob')$ where $\langle p, m\rangle Y \xrightarrow{x} \langle p', m'\rangle \alpha$ iff $pY \xrightarrow{x} p'\alpha$ in Δ and $m \xrightarrow{pY} m'$ in $\mathcal{M}$. For every run w of $\Delta_{\mathcal{M}}$, let $\inf(w)$ be the set of all $m \in M$ for which there are infinitely many $i \in \mathbb{N}$ such that $w(i) = \langle p, m\rangle \alpha$, where $p \in Q$ and $\alpha \in \Gamma^*$. A run w of $\Delta_{\mathcal{M}}$ is *accepting* if $\inf(w) \in \mathcal{F}$. For every $\langle p, m\rangle X$, let $Run(\langle p, m\rangle X, Acc)$ be the set of all $w \in IRun(\langle p, m\rangle X)$ which are accepting. Obviously, for each pX we have that $\mathcal{P}(Run(pX, \mathcal{M})) = \mathcal{P}(Run(\langle p, m_I\rangle X, Acc))$. Therefore, in the rest of this section we consider just the problem whether $\mathcal{P}(Run(\langle p, m\rangle X, Acc)) \sim \rho$.

Definition 3.2. *Let w be an infinite run of $\Delta_{\mathcal{M}}$, and let $Stack(w(i))$ be the stack content of $w(i)$ for each $i \in \mathbb{N}_0$. For each $i \in \mathbb{N}$ we define the i^{th}* minimum *of w, denoted $\min_i(w)$, inductively as follows:*

- $\min_1(w) = w(k)$, *where $k \in \mathbb{N}_0$ is the least number such that for each $k' \geq k$ we have that $|Stack(w(k'))| \geq |Stack(w(k))|$.*
- $\min_{i+1}(w) = \min_1(w_{\ell+1})$, *where $\min_i(w) = w(\ell)$.*

We say that $m \in M$ is seen *at $\min_i(w)$ if either $i = 1$ and $\min_1(w)$ is of the form $\langle q, m\rangle \alpha$, or $i > 1$ and w visits a configuration of the form $\langle q, m\rangle \alpha$ between $\min_{i-1}(w)$ and $\min_i(w)$ (where $\min_{i-1}(w)$ is not included).*

For every $\langle p, m\rangle X \in \mathcal{H}(\Delta_{\mathcal{M}})$ and every $i \in \mathbb{N}$ we define a random variable $V^{(i)}_{\langle p,m\rangle X}$ over $Run(\langle p, m\rangle X)$ as follows: The set $\mathcal{V}$ of possible values of these variables is $\{(\langle q, r\rangle Y, S) \mid \langle q, r\rangle Y \in \mathcal{H}(\Delta_{\mathcal{M}}), S \subseteq M\} \cup \{\bot\}$. For a given $w \in Run(\langle p, m\rangle X)$, $V^{(i)}_{\langle p,m\rangle X}(w)$ is determined as follows:

- if w is finite, then $V^{(i)}_{\langle p,m\rangle X}(w) = \bot$;

- if w is infinite, the head of $\min_i(w)$ is $\langle q, r\rangle Y$, and the set of all $s \in M$ that are seen at $\min_i(w)$ is S, then $V^{(i)}_{\langle p,m\rangle X}(w) = (\langle q, r\rangle Y, S)$.

Lemma 3.3 (cf. Lemma 5.6. in [EKM04]). *For every $\langle p, m\rangle X \in \mathcal{H}(\Delta_{\mathcal{M}})$, every $n \in \mathbb{N}$, and all $v_1, \cdots, v_n \in \mathcal{V}$, the probability $\mathcal{P}(V^{(1)}_{\langle p,m\rangle X}{=}v_1 \wedge \cdots \wedge V^{(n)}_{\langle p,m\rangle X}{=}v_n)$ exists, i.e., the set of all $w \in Run(\langle p, m\rangle X)$ which satisfy this condition is $\mathcal{P}$-measurable.*

Lemma 3.4 (cf. Lemma 5.7. and 5.8. in [EKM04]). *Let $\langle p, m\rangle X \in \mathcal{H}(\Delta_{\mathcal{M}})$, $n \in \mathbb{N}$, and $v_1, \cdots, v_n \in \mathcal{V}$, where $v_{n-1} = (\langle q, r\rangle Y, S)$. If $\mathcal{P}(V^{(1)}_{\langle p,m\rangle X}{=}v_1 \wedge \cdots \wedge V^{(n-1)}_{\langle p,m\rangle X}{=}v_{n-1})$ is non-zero, then*

$$\begin{aligned} & \mathcal{P}(V^{(n)}_{\langle p,m\rangle X}{=}v_n \mid V^{(n-1)}_{\langle p,m\rangle X}{=}v_{n-1} \wedge \cdots \wedge V^{(1)}_{\langle p,m\rangle X}{=}v_1) \quad = \\ = \quad & \mathcal{P}(V^{(n)}_{\langle p,m\rangle X}{=}v_n \mid V^{(n-1)}_{\langle p,m\rangle X}{=}v_{n-1}) \quad = \quad \mathcal{P}(V^{(2)}_{\langle q,r\rangle Y}{=}v_n \mid V^{(1)}_{\langle q,r\rangle Y}{=}(\langle q, r\rangle Y, \{r\})) \end{aligned}$$

Now we can define a finite Markov chain $\mathsf{F}(\Delta, \mathcal{M})$, where the set of states is

$$\{\bot\} \cup \mathcal{H}(\Delta_{\mathcal{M}}) \cup \{(\langle p, m\rangle X, S) \in \mathcal{V} \mid \mathcal{P}(V^{(1)}_{\langle p,m\rangle X}{=}(\langle p, m\rangle X, \{m\})) > 0\}$$

and transition probabilities are defined as follows:

- $Prob(\bot \to \bot) = 1$
- $Prob(\langle p, m\rangle X \to (\langle q, r\rangle Y, S)) \quad = \quad \mathcal{P}(V^{(1)}_{\langle p,m\rangle X}{=}(\langle q, r\rangle Y, S))$
- $Prob(\langle p, m\rangle X \to \bot) \quad = \quad \mathcal{P}(V^{(1)}_{\langle p,m\rangle X}{=}\bot)$
- $Prob((\langle p, m\rangle X, S) \to (\langle q, r\rangle Y, T)) = \mathcal{P}(V^{(2)}_{\langle p,m\rangle X}{=}(\langle q, r\rangle Y, T) \mid V^{(1)}_{\langle p,m\rangle X}{=}(\langle p, m\rangle X, \{m\}))$

A *trajectory* in $\mathsf{F}(\Delta, \mathcal{M})$ is an infinite sequence $d_0, d_1, \cdots$ of states of $\mathsf{F}(\Delta, \mathcal{M})$ such that $Prob(d_i \to d_{i+1}) > 0$ for each $i \in \mathbb{N}_0$. For each $w \in Run(\langle p, m\rangle X)$ we define the *footprint* of w, which is the sequence $V^{(1)}_{\langle p,m\rangle X}(w), V^{(2)}_{\langle p,m\rangle X}(w), \cdots$. A run is *good* if its footprint is a trajectory in $\mathsf{F}(\Delta, \mathcal{M})$. Observe that there can be runs which are not good. However, by using similar arguments as in [EKM04], one can easily show that the total probabilistic measure of all such runs is zero. So, we can safely restrict ourselves to good runs. The next step is to realize that since $\mathsf{F}(\Delta, \mathcal{M})$ is finite, a trajectory of $\mathsf{F}(\Delta, \mathcal{M})$ hits a state of some bottom strongly connected component C of $\mathsf{F}(\Delta, \mathcal{M})$ with probability one, and from that point on *each* state of C will be visited infinitely often with probability one. This is a standard observation which holds for an arbitrary finite-state Markov chain. We use this claim in the following sequence of observations about good runs.

Let BC be the set of all bottom strongly connected components of $\mathsf{F}(\Delta, \mathcal{M})$. Each $C \in BC$ is either *accepting* or *rejecting*, depending on whether the set $\bigcup_{(\langle q,r\rangle Y,S)\in C} S$ belongs

to $\mathcal{F}$ or not, respectively (remember that $\mathcal{F}$ is the set of accepting sets of states in $\mathcal{M}$). Note that $\{\bot\}$ is also a bottom strongly connected component which is rejecting. Now let $C \in BC$. We say that a good run $w \in Run(\langle p, m\rangle X)$ *hits* C if the footprint of w hits C. Let $Run(\langle p, m\rangle X, C)$ be the set of all runs that are good and hit C. Obviously, $\mathcal{P}(Run(\langle p, m\rangle X, C))$ is equal to the probability that a trajectory in $F(\Delta, \mathcal{M})$ initiated in $\langle p, m\rangle X$ hits C. Hence, $\sum_{C \in BC} \mathcal{P}(Run(\langle p, m\rangle X, C)) = 1$ due to the above observation about $F(\Delta, \mathcal{M})$. Moreover, the conditional probability that a good run $w \in Run(\langle p, m\rangle X)$ is accepting on the hypothesis that w hits C is equal either to 1 or 0, depending on whether C is accepting or rejecting, respectively. Here we used the second part of the observation about $F(\Delta, \mathcal{M})$. To sum up, we obtain the following:

Lemma 3.5. *$\mathcal{P}(Run(\langle p, m\rangle X, Acc))$ is equal to the probability that a trajectory in $F(\Delta, \mathcal{M})$ initiated in $\langle p, m\rangle X$ hits an accepting bottom strongly connected component of $F(\Delta, \mathcal{M})$.*

Hence, if the transition probabilities of $F(\Delta, \mathcal{M})$ were known, we could compute $\mathcal{P}(Run(\langle p, m\rangle X, Acc))$ just by applying, e.g., the results of [CY95]. However, at the moment we only know that these probabilities *exist*. The missing piece of puzzle is provided in our next lemma.

Lemma 3.6. *Let d, e be states of $F(\Delta, \mathcal{M})$, $\sim \in \{\leq, <, \geq, >\}$, and y be a first-order variable. Then there effectively exists a formula $\varphi^{\sim y}(d \to e)$ of $(\mathbb{R}, +, *, \leq)$ such that:*

- *The variable y is the only free variable of $\varphi^{\sim y}(d \to e)$. Moreover, for each $\rho \in [0, 1]$ we have that the formula $\varphi^{\sim y}(d \to e)[\rho/y]$ holds iff $Prob(d \to e) \sim \rho$.*
- *The formula $\varphi^{\sim y}(d \to e)$ is constructible in time which is polynomial in the size of Δ and exponential in the size of $\mathcal{M}$, and the quantifier alternation depth of $\varphi^{\sim y}(d \to e)$ is fixed.*

Moreover, the problem if $Prob(d \to e) > 0$ is decidable in space which is polynomial in the size of Δ and exponential in the size of $\mathcal{M}$ (more precisely, the space complexity is exponential only in the number of states of $\mathcal{M}$).

Proof. We will use the results on random walks which were recalled in Section 2. We show that there effectively exists an expression E consisting of some probabilities of the form $\mathcal{P}(pX, \mathcal{C}_1 \mathcal{U} \mathcal{C}_2)$, where $\mathcal{C}_1, \mathcal{C}_2$ are simple sets of configurations of Δ, which are combined using summation and multiplication, such that $Prob(d \to e) = E$. Since the $\mathcal{P}(pX, \mathcal{C}_1 \mathcal{U} \mathcal{C}_2)$ probabilities which were used in E are themselves definable

in $(\mathbb{R}, +, *, \leq)$ in the way indicated in Section 2, the formula $\varphi^{\sim y}(d \to e)$ is obtained essentially by writing down the definitions of those probabilities which appear in E, and stipulating that $E \sim y$.

Let start by considering the case when d is of the form $\langle p, m\rangle X$ and $e = \bot$. Then, by definition of $F(\Delta, \mathcal{M})$, $Prob(d \to e)$ is the probability that a run initiated in $\langle p, m\rangle X$ is finite. That is, $Prob(d \to e) = \mathcal{P}(Run(\langle p, m\rangle X, \mathtt{tt}\,\mathcal{U}\, dead))$, where *dead* is a simple set of configurations determined by all heads of the form $\langle q, n\rangle \varepsilon$ together with all heads of the form $\langle q, n\rangle Y$ such that the configuration $\langle q, n\rangle Y$ has no outgoing transitions. So, we are done.

If $d = \langle p, m\rangle X$ and $e = (\langle q, r\rangle Y, S))$, then $Prob(d \to e) = \mathcal{P}(V^{(1)}_{\langle p,m\rangle X} = (\langle q, r\rangle Y, S))$. It follows directly from the definition of $V^{(1)}_{\langle p,m\rangle X}$ that this probability can be non-zero only if $\langle q, r\rangle Y = \langle p, m\rangle X$ and $S = \{m\}$. In all other cases, we define $\varphi^{\sim y}(d \to e)$ to be $y{=}0$. Now it suffices to realize that $\mathcal{P}(V^{(1)}_{\langle p,m\rangle X} = (\langle p, m\rangle X, \{m\}))$ is actually the probability that a run initiated in $\langle p, m\rangle X$ is infinite. That is, $Prob(d \to e) = 1 - \mathcal{P}(Run(\langle p, m\rangle X, \mathtt{tt}\,\mathcal{U}\, dead))$.

Finally, let us consider the case when $d = (\langle p, m\rangle X, S)$ and $e = (\langle q, r\rangle Y, T))$. By definition, $Prob(d \to e)$ is equal to $\mathcal{P}(V^{(2)}_{\langle p,m\rangle X} = (\langle q, r\rangle Y, T)) \mid V^{(1)}_{\langle p,m\rangle X} = (\langle p, m\rangle X, \{m\}))$. This probability cannot be directly expressed in terms of random walks over configurations of $\Delta_{\mathcal{M}}$. However, it suffices to consider another pPDA Δ' constructed from Δ and $\mathcal{M}$ as follows: $\Delta' = (Q{\times}M{\times}2^M, \Gamma, \delta', Prob')$, where $\langle p, m, S\rangle Y \xrightarrow{x} \langle p', m', S'\rangle \alpha$ iff $pY \xrightarrow{x} p'\alpha$ in Δ, $m \xrightarrow{pY} m'$ in $\mathcal{M}$, and $S' = S \cup \{m'\}$. Intuitively, configurations of Δ' "remember" all states of $\mathcal{M}$ that have been visited so far. Now it is straightforward to check that $\mathcal{P}(V^{(2)}_{\langle p,m\rangle X} = (\langle q, r\rangle Y, T)) \mid V^{(1)}_{\langle p,m\rangle X} = (\langle p, m\rangle X, \{m\}))$ is equal to the conditional probability that the head of the second minimum of a run initiated in the configuration $\langle p, m, \emptyset\rangle X$ of Δ' is $\langle q, r, T\rangle Y$ on the hypothesis that the head of the first minimum of this run is $\langle p, m, \emptyset\rangle X$. Now we can apply the result of [EKM04] which says that this probability is effectively expressible from probabilities of the form $\mathcal{P}(sX, \mathcal{C}_1\,\mathcal{U}\,\mathcal{C}_2)$, where s is a control state of Δ' and $\mathcal{C}_1, \mathcal{C}_2$ are simple sets of configurations of Δ'. In this case, the formula $\varphi^{\sim y}(d \to e)$ is constructible in time which is polynomial in the size of Δ', that is, in time which is polynomial in the size of Δ and exponential in the size of $\mathcal{M}$.

Since $Prob(d \to e)$ is expressible from probabilities of the form $\mathcal{P}(pX, \mathcal{C}_1\,\mathcal{U}\,\mathcal{C}_2)$ using only multiplication and summation, the problem whether $Prob(d \to e) > 0$ can be decided just by determining whether these basic probabilities are non-zero. Due to Lemma 2.4 and the previous complexity estimations we can conclude that the whole

procedure requires space which is polynomial in the size of Δ and exponential in the size of $\mathcal{M}$. □

Theorem 3.7. *The quantitative model-checking problem for pPDA processes and ω-regular properties represented by Muller automata is in **2-EXPTIME**, and the qualitative variant of this problem is in **EXPSPACE**.*

Proof. In the qualitative case (i.e., when we are to decide whether $\mathcal{P}(Run(\langle p, m\rangle X, Acc))$ is equal to 1 or 0), it suffices to find out whether all of the bottom strongly connected components of $\mathsf{F}(\Delta, \mathcal{M})$ that are reachable from $\langle p, m\rangle X$ are accepting or rejecting, respectively. Realize that the size of $\mathsf{F}(\Delta, \mathcal{M})$ is polynomial in the size of Δ and exponential in the size of $\mathcal{M}$. For each pair of states d, e of $\mathsf{F}(\Delta, \mathcal{M})$ we decide whether $Prob(d \rightarrow e) > 0$ by checking the validity of the formula $\varphi^{\leq 0}(d \rightarrow e)$ of Lemma 3.6 and negating the answer. Since the existential fragment of $(\mathbb{R}, +, *, \leq)$ is decidable in polynomial space, this is achievable in space which is polynomial in the size of Δ and exponential in the size of $\mathcal{M}$. Partitioning $\mathsf{F}(\Delta, \mathcal{M})$ into strongly conected components and checking the above mentioned reachability question can be done in time which is polynomial in the size of $\mathsf{F}(\Delta, \mathcal{M})$. To sum up, the qualitative case is solvable in space which is polynomial in the size of Δ and exponential in the size of $\mathcal{M}$.

The general question (i.e., whether $\mathcal{P}(Run(\langle p, m\rangle X, Acc)) \sim \rho$) can be decided by constructing a closed formula Φ of $(\mathbb{R}, +, *, \leq)$ such that $\mathcal{P}(Run(\langle p, m\rangle X, Acc)) \sim \rho$ iff Φ holds. This is done in the very same way as in [EKM04], using Lemma 3.6 appropriately. The formula Φ is constructible in time which is polynomial in Δ and exponential in $\mathcal{M}$. Since the universal and existential quantifiers are alternated in Φ only to a fixed depth, we can apply the result of [Gri88] and conclude that the validity of Φ is decidable in time which is exponential in the size of Δ and doubly exponential in the size of $\mathcal{M}$. □

Corollary 3.8. *The quantitative model-checking problem for pPDA processes and ω-regular properties represented by Büchi automata is in **3-EXPTIME**, and the qualitative variant of this problem is in **2-EXPSPACE**.*

4 Model-Checking PCTL, PCTL*, and PECTL* Properties

We start by proving that model-checking PCTL is undecidable for pPDA processes, and model-checking PCTL$^+$ is undecidable for pBPA processes.

A *Minsky machine* with two counters is a finite sequence $\mathcal{C}$ of labeled instructions ℓ_1:$inst_1, \cdots, \ell_n$:$inst_n$, where $n \geq 1$, $inst_n = \texttt{halt}$, and for every $1 \leq i < n$, the instruction $inst_i$ is of one of the following two types:

Type I. $\quad c_r := c_r + 1;\ \texttt{goto}\ \ell_j$

Type II. $\quad \text{if } c_r = 0\ \texttt{then goto}\ \ell_j\ \texttt{else}\ c_r := c_r - 1;\ \texttt{goto}\ \ell_k$

Here $r \in \{1, 2\}$ is a counter index. A *configuration* of $\mathcal{C}$ is a triple (ℓ_i, v_1, v_2), where $1 \leq i \leq n$ and $v_1, v_2 \in \mathbb{N}_0$ are counter values. Each configuration (ℓ_i, v_1, v_2) has a unique *successor* which is the configuration obtained by performing $inst_i$ on (ℓ_i, v_1, v_2). The *halting problem* for Minsky machines with two counters initialized to zero, i.e., the question whether $(\ell_1, 0, 0)$ eventually reaches a configuration of the form (ℓ_n, v_1, v_2), where $v_1, v_2 \in \mathbb{N}_0$, is undecidable [Min67].

Our aim is to reduce the halting problem for Minsky machines to the PCTL model checking problem for pPDA.

Let $\mathcal{C}$ be a Minsky machine. We construct a pPDA system Δ, a process $p\alpha$ of Δ, and a PCTL formula ψ such that $\mathcal{C}$ halts iff $p\alpha \models \psi$. The formula ψ looks as follows:

$$\psi \ \equiv\ \mathcal{P}^{>0}((\mathit{check} \Rightarrow (\varphi_{\mathit{state}} \wedge \varphi_{\mathit{zero}} \wedge \varphi_{\mathit{count}}))\ \mathcal{U}\ \mathit{halt})$$

Here *check* and *halt* are atomic propositions, φ_{state} and φ_{zero} are qualitative formulae with just one $\mathcal{U}$ operator, and φ_{count} is a quantitative formula with just one $\mathcal{U}$ operator. So, φ_{count} is the only non-qualitative subformula in ψ. The stack content of the initial process $p\alpha$ corresponds to the initial configuration of $\mathcal{C}$. In general, a configuration (ℓ_i, v_1, v_2) is represented by the sequence $\ell_i A^{v_1} B^{v_2}$ of stack symbols, and individual configurations are separated by the # marker.

Starting from $p\alpha$, Δ tries to "guess" the successor configuration of $\mathcal{C}$ by pushing a sequence of stack symbols of the form $\ell_j A^{v_1} B^{v_2}$#. The transitions of Δ are arranged so that only strings of this syntactical form can be pushed. Transition probabilities do not matter here, the only important thing is that the "right" configuration can be guessed with a non-zero probability. After guessing the configuration (i.e., after pushing the symbol ℓ_j), Δ inevitably pushes one of the special "checking" symbols of the form (ℓ_i, ℓ_j, r, d), where $1 \leq i \leq n$, $r \in \{1, 2\}$ is a counter index, and $d \in \{-1, 0, 1\}$ a counter change (note that the previously pushed ℓ_j is in the second component of the checking symbol). An intuitive meaning of checking symbols is explained later. Let us just note that checking symbols

correspond to instructions of $\mathcal{C}$ and hence not all tuples of the form (ℓ_i, ℓ_j, r, d) are necessarily checking symbols. Still, there can be several checking symbols with the same ℓ_j in the second component, and Δ can freely choose among them. Actually, the checking symbol is pushed *together* with ℓ_j, and hence the guessing phase ends in a "checking configuration" where the stack looks as follows: $(\ell_i, \ell_j, r, d)\ell_j A^{\nu_1} B^{\nu_2} \# \ldots$. The atomic proposition *check* is valid in exactly all checking configurations (i.e., configurations with a checking symbol on top of the stack), and the proposition *halt* is valid in exactly those configurations where ℓ_n (i.e., the label of `halt`) is on top of the stack.

From a checking configuration, Δ can either pop the checking symbol (note that the symbol ℓ_j appears at the top of the stack at this moment) and go on with guessing another configuration of $\mathcal{C}$, or perform other transitions so that the subformulae φ_{state}, φ_{zero}, and φ_{count} are (possibly) satisfied. Hence, the formula ψ says that there is a finite sequence of transitions from $p\alpha$ leading to a "halting" configuration along which all checking configurations satisfy the formulae φ_{state}, φ_{zero}, and φ_{count}. As can be expected, these three subformulae together say that the configuration of $\mathcal{C}$ just pushed to the stack is the successor of the configuration which was pushed previously. Let us discuss this part in greater detail.

First, let us clarify the meaning of checking symbols. Intuitively, each checking symbol corresponds to some computational step of $\mathcal{C}$. More precisely, the set of all checking symbols is the least set $\mathcal{T}$ such that for every $1 \leq i \leq n$ we have that

- if $inst_i \equiv c_r := c_r + 1;$ `goto` ℓ_j, then $(\ell_i, \ell_j, r, 1) \in \mathcal{T}$;
- if $inst_i \equiv$ `if` $c_r = 0$ `then goto` ℓ_j `else` $c_r := c_r - 1;$ `goto` ℓ_k, then $(\ell_i, \ell_j, r, 0), (\ell_i, \ell_k, r, -1) \in \mathcal{T}$.

Note that the checking symbol (ℓ_i, ℓ_j, r, d) which is pushed together with ℓ_j at the end of guessing phase is chosen freely. So, this symbol can also be chosen "badly" in the sense that ℓ_i is not the label of the previously pushed configuration, or the wrong branch of a Type II instruction is selected.

The formula φ_{state} intuitively says that we have chosen the "right" ℓ_i, and the subformula φ_{zero} says that if the checking symbol (ℓ_i, ℓ_j, r, d) claims the use of a Type II instruction and the counter c_r was supposed to be zero (i.e., $d = 0$), then the previously pushed configuration of $\mathcal{C}$ indeed has zero in the respective counter. In other words, φ_{zero} verifies that the right branch of a Type II instruction was selected.

The most interesting part is the subformula φ_{count}, which says that the counter values in the current and the previous configuration have changed accordingly to (ℓ_i, ℓ_j, r, d). For example, if $r = 0$ and $d = -1$, then the subformula φ_{count} is valid in the considered checking configuration iff the first counter was changed by -1 and the second counter remained unchanged.

To get some intuition on how this can be implemented, let us consider a simplified version of this problem. Let us assume that we have a configuration of the form $pA^m\#A^n\#$. Our aim is to set up the transitions of $pA^m\#A^n\#$ and to construct a PCTL formula φ so that $pA^m\#A^n\# \models \varphi$ iff $m = n$ (this indicates how to check if a counter remains unchanged). Let

$$\begin{array}{llllll} pA \xrightarrow{1/2} qA, & qA \xrightarrow{1} q\varepsilon, & rA \xrightarrow{1/2} sA, & tA \xrightarrow{1/2} t\varepsilon, & sA \xrightarrow{1} sA, \\ pA \xrightarrow{1/2} tA, & q\# \xrightarrow{1} r\varepsilon, & rA \xrightarrow{1/2} r\varepsilon, & tA \xrightarrow{1/2} uA, & uA \xrightarrow{1} uA \\ & & & t\# \xrightarrow{1} sA, & \end{array}$$

By inspecting possible runs of $pA^m\#A^n\#$, one can easily confirm that the probability that a run of $pA^m\#A^n\#$ hits a configuration having sA as its head is exactly

$$\frac{1}{2} \cdot (1 - \frac{1}{2^n}) + \frac{1}{2} \cdot \frac{1}{2^m} \quad = \quad \frac{1}{2} - \frac{1}{2^{n+1}} + \frac{1}{2^{m+1}}$$

Let p_{sA} be an atomic proposition which is valid in (exactly) all configurations having sA as their head. Then $pA^m\#A^n\# \models \mathcal{P}^{=\frac{1}{2}}(tt\,\mathcal{U}\,p_{sA})$ iff $m = n$.

One can argue that formulae where some probability is required to be *equal* to some value are seldom used in practice. However, it is easy to modify the proof so that for every subformula of the form $\mathcal{P}^{\sim\rho}\varphi$ which is employed in the proof we have that $\sim$ is $>$ and ρ is a "simple" rational like $1/2$ or $1/4$.

Theorem 4.1. *The model-checking problem for pPDA processes and the logic PCTL is undecidable. Moreover, the undecidability result holds even for the fragment of PCTL where the nesting depth of $\mathcal{U}$ is at most two, and for all subformulae of the form $\mathcal{P}^{\sim\rho}\varphi$ we have that $\sim$ is $>$.*

The proof of Theorem 4.1 does not carry over to pBPA processes. The decidability of PCTL for pBPA processes is one of the challenges which are left open for future work. Nevertheless, we were able to show that model-checking PCTL$^+$ (and in fact a simple fragment of this logic) is undecidable even for pBPA. The structure of the construction is similar as in Theorem 4.1, but the proof contains new tricks invented specifically for pBPA. In particular, the consistency of counter values in consecutive configurations is verified somewhat differently. This is the only place where we use the expressive power of PCTL$^+$.

Theorem 4.2. *The model-checking problem for pBPA processes and the logic PCTL$^+$ is undecidable. More precisely, the undecidability result holds even for a fragment of PCTL$^+$ where the nesting depth of $\mathcal{U}$ is at most two, and for all subformulae of the form $\mathcal{P}^{\sim\rho}\varphi$ we have that $\sim$ is $>$.*

The formal proofs of Theorem 4.1 and Theorem 4.2 are given in Appendix.

Finally, let us note that our undecidability result is tight with respect to the nesting depth of $\mathcal{U}$. The fragment of PCTL where the $\mathcal{U}$ operators are not nested (and the $\mathcal{X}$ operators can be nested to an arbitrary depth) is decidable by applying the results of [EKM04]. In our undecidability proof we use a PCTL formula where the nesting depth of $\mathcal{U}$ is 2 (PCTL formulae where the $\mathcal{U}$ operators are not nested have the nesting depth 1).

4.1 Model-checking qPECTL*

Now we prove that the model-checking problem for pPDA and the logic qPECTL* is decidable and belongs to **2-EXPSPACE**. For the logic qPCTL, our algorithm only needs singly exponential space.

Let us fix a pPDA $\Delta = (Q, \Gamma, \delta, \mathit{Prob})$, qPECTL* formula τ, and a simple valuation ν. The symbol $\mathit{Cl}(\tau)$ denotes the set of all subformulae of τ, and $\mathit{Acl}(\tau) \subseteq \mathit{Cl}(\tau)$ is the subset of all "automata subformulae" of the form $\mathcal{P}^{=x}\mathcal{B}$.

Let $\varphi \equiv \mathcal{P}^{=x}\mathcal{B} \in \mathit{Acl}(\tau)$ where $\mathcal{B}$ is a Büchi automaton over an alphabet $\Sigma_\varphi = 2^{\{\Phi_1,\ldots,\Phi_n\}}$. Then there is a (deterministic) Muller automaton $\mathcal{M}_\varphi = (\Sigma_\varphi, M_\varphi, \rho_\varphi, m^I_\varphi, \mathcal{F}_\varphi)$ whose size is at most exponential in the size of $\mathcal{B}$ such that $L(\mathcal{M}_\varphi) = L(\mathcal{B})$. In our constructions we use $\mathcal{M}_\varphi$ instead of $\mathcal{B}$.

The intuition behind our proof is that we extend each configuration of Δ with some additional information that allows to determine the (in)validity of each subformula of τ in a given configuration just by inspecting the head of the configuration. Our algorithm computes a sequence of *extensions of* Δ that are obtained from Δ by augmenting stack symbols and transition rules with some information about subformulae of τ. These extensions are formally introduced in our next definition. For notation convenience, we define $\mathit{St} = \Pi_{\varphi\in\mathit{Acl}(\tau)} 2^{Q\times M_\varphi}$. For every $\nu \in \mathit{St}$, the projection of ν onto a given $\varphi \in \mathit{Acl}(\tau)$ is denoted $\nu(\varphi)$. Note that $\nu(\varphi)$ is a set of pairs of the form (q, m), where $q \in Q$ and $m \in M_\varphi$.

Definition 4.3. *We say that a pPDA* $\Delta' = (Q, \Gamma', \delta', Prob')$ *is an* extension of Δ *if and only if* $\Gamma' = St \times \Gamma \times St$ *(elements of* Γ' *are written as* (uXv)*, where* $u, v \in St$ *and* $X \in \Gamma$*), and the outgoing transitions of every* $p(uXv) \in Q \times \Gamma'$ *satisfy the following:*

1. *if* $pX \xrightarrow{x} q\varepsilon$*, then* $p(uXv) \xrightarrow{x} q\varepsilon$*;*
2. *if* $pX \xrightarrow{x} qY$*, then there is a* unique $z \in St$ *such that* $p(uXv) \xrightarrow{x} q(zYv)$*;*
3. *if* $pX \xrightarrow{x} qYZ$*, then there are* unique $z, w \in St$ *such that* $p(uXv) \xrightarrow{x} q(zYw)(wZv)$*;*
4. $p(uXv)$ *has no other outgoing transitions.*

Note that due to 2. and 3., a given Δ can have many extensions. However, all of these extensions have the same set of control states and the same stack alphabet. Moreover, the part of $\mathcal{T}_{\Delta'}$ which is reachable from a configuration $p(u_1X_1v_1)\cdots(u_nX_nv_n)$ is isomorphic to the part of $\mathcal{T}_\Delta$ reachable from the configuration $pX_1\cdots X_n$.

Definition 4.4. *Let* $\Delta' = (Q, \Gamma', \delta', Prob')$ *be an extension of* Δ*. For each* $\varphi \in Cl(\tau)$ *we define a set* $\mathcal{C}_\varphi \subseteq Q \times \Gamma'$ *inductively as follows:*

- *if* $\varphi = a$ *where* $a \in Ap$*, then* $\mathcal{C}_\varphi = \{p(uXv) \mid pX \in f_v(a)$ *and* $u, v \in St\}$
- *if* $\varphi = \psi \wedge \xi$*, then* $\mathcal{C}_\varphi = \mathcal{C}_\psi \cap \mathcal{C}_\xi$
- *if* $\varphi = \neg\psi$*, then* $\mathcal{C}_\varphi = (Q \times \Gamma') \setminus \mathcal{C}_\psi$
- *if* $\varphi = \mathcal{P}^{=x}\mathcal{B}$*, then* $\mathcal{C}_\varphi = \{p(uXv) \mid u, v \in St$ *and* $(p, m^I_\varphi) \in u(\varphi)\}$

For each $\varphi \in Acl(\tau)$ *we define a Muller automaton* $\mathcal{M}'_\varphi = (\Sigma'_\varphi, M_\varphi, \rho'_\varphi, m^I_\varphi, \mathcal{F}_\varphi)$*, which is a modification of the automaton* $\mathcal{M}_\varphi$*, as follows:* $\Sigma'_\varphi = Q \times \Gamma'$*, and* $m \xrightarrow{h} m'$ *is a transition of* ρ'_φ *iff there is* $A \in \Sigma_\varphi$ *such that* $m \xrightarrow{A} m'$ *is a transition of* ρ_φ *and* $h \in (\bigcap_{\psi\in A} \mathcal{C}_\psi) \setminus \bigcup_{\psi\notin A} \mathcal{C}_\psi$*. Note that* $\mathcal{M}'_\varphi$ *is again deterministic.*

Let Δ' be an extension of Δ. The symbol $[s, p(uXv)\bullet]_\varphi$ denotes the probability that a run of $Run(p(uXv))$ is accepted by $\mathcal{M}'_\varphi$ where the initial state of $\mathcal{M}'_\varphi$ is changed to s. Furthermore, the symbol $[s, p(uXv)q, t]_\varphi$ denotes the probability that a run w of $Run(p(uXv))$ hits the configuration $q\varepsilon$, i.e., w is of the form $w'\,q\varepsilon$, so that $\mathcal{M}'_\varphi$ initiated in s moves to t after reading the heads of all configurations in w'.

Intuitively, the sets $\mathcal{C}_\varphi$ are supposed to encode exactly those configurations where φ holds (the information which is relevant for the (in)validity of φ should have been

accumulated in the symbol at the top of the stack). However, this works only under some "consistency" assumptions, which are formalized in our next definition (see also Lemma 4.6 below).

Definition 4.5. *Let $\varphi \in Acl(\tau)$ and let Δ' be an extension of Δ. We say that a symbol $(uXv) \in \Gamma'$ is* φ-consistent in Δ' *iff the following conditions are satisfied:*

- *if $\varphi \equiv \mathcal{P}^{=1}\mathcal{B}$, then $u(\varphi) = \{(p,s) \mid [s, p(uXv)\bullet]_\varphi + \sum_{(q,t)\in\nu(\varphi)}[s, p(uXv)q, t]_\varphi = 1\}$*
- *if $\varphi \equiv \mathcal{P}^{=0}\mathcal{B}$, then $u(\varphi) = \{(p,s) \mid [s, p(uXv)\bullet]_\varphi + \sum_{(q,t)\notin\nu(\varphi)}[s, p(uXv)q, t]_\varphi = 0\}$*

We say that a configuration $p(u_1X_1v_1)\cdots(u_nX_nv_n)$ is φ-consistent in Δ' iff $(u_iX_iv_i)$ is φ-consistent in Δ' for every $1 \le i \le n$, and $v_i = u_{i+1}$ for every $1 \le i < n$.

An extension Δ' of Δ is φ-consistent iff for all transitions of the form $p(uXv) \xrightarrow{x} q(zYv)$ and $p(uXv) \xrightarrow{x} q(zYw)(wZv)$ of Δ' we have that $q(zYv)$ and $q(zYw)(wZv)$ are φ-consistent in Δ', respectively.

It is important to realize that the conditions of Definition 4.5 are effectively verifiable, because, e.g., the condition $[s, p(uXv)\bullet]_\varphi + \sum_{(q,t)\in\nu(\varphi)}[s, p(uXv)q, t]_\varphi = 1$ can effectively be translated into $(\mathbb{R}, +, *, \le)$ using the construction of Theorem 3.7 and the results on random walks of [EKM04] which were recalled in Section 2.

A $\nu \in St$ is *terminal* iff for each $\varphi \in Acl(\tau)$ we have that if $\varphi = \mathcal{P}^{=1}\mathcal{B}$ then $\nu(\varphi) = \emptyset$, and if $\varphi = \mathcal{P}^{=0}\mathcal{B}$ then $\nu(\varphi) = Q \times M_\varphi$.

Lemma 4.6. *Let $\varphi \in Cl(\tau)$, and let Δ' be an extension of Δ which is ψ-consistent for all $\psi \in Acl(\varphi)$. Let $p(u_1X_1v_1)\cdots(u_nX_nv_n)$ (where $n \ge 1$) be a configuration of Δ' which is ψ-consistent in Δ' for each $\psi \in Acl(\varphi)$, and where v_n is terminal. Then $pX_1\cdots X_n \models \varphi$ iff $p(u_1X_1v_1) \in \mathcal{C}_\varphi$.*

Proof. First we fix some notation. Let $\mathcal{M} = (\Sigma, M, \rho, m^I, \mathcal{F})$ be a Muller automaton. For every $s \in M$, the symbol $\mathcal{M}^s$ denotes the Muller automaton obtained from $\mathcal{M}$ by changing the initial state to s. Moreover, we define for all $s, t \in M$ a deterministic finite automaton over finite words $\mathcal{M}^{s\to t} = (\Sigma, M, \rho, s, \{t\})$. Given two runs w and w' we denote $w\,w'$ the concatenation of the runs. If a run w is of the form $p_1\alpha_1, p_2\alpha_2\cdots$ and $\beta \in \Gamma^*$, then we denote $w \odot \beta$ the run of the form $p_1\alpha_1\beta, p_2\alpha_2\beta, \cdots$. The notation is extended to sets of runs in the obvious way.

We prove the following statement by induction on the structure of φ. The cases when φ is an atomic proposition or a Boolean combination of simpler formulae are clear and hence we concentrate only on automata connectives. We proceed in two steps.

Step 1: Suppose that $\varphi = \mathcal{P}^{=x}\mathcal{B}$. The following equation is obtained by applying induction hypothesis and the definition of $\mathcal{M}'_\varphi$.

$$\begin{array}{ll}
\mathcal{P}(\{w \in \mathit{IRun}(pX_1 \cdots X_n) \mid w_{\mathcal{B}} \in \text{Ł}(\mathcal{M}^s_\varphi)\}) & = \\
\\
\mathcal{P}(\{w \in \mathit{IRun}(pX_1) \odot X_2 \cdots X_n \mid w_{\mathcal{B}} \in \text{Ł}(\mathcal{M}^s_\varphi)\}) & + \\
\sum_{(q,t)\in Q\times M_\varphi} \mathcal{P}(\{w\, qX_2 \cdots X_n \in \mathit{Run}(pX_1) \odot X_2 \cdots X_n \mid w_{\mathcal{B}} \in \text{Ł}(\mathcal{M}^{s\longrightarrow t}_\varphi)\}) & \cdot \\
\mathcal{P}(\{w \in \mathit{IRun}(qX_2 \cdots X_n) \mid w_{\mathcal{B}} \in \text{Ł}(\mathcal{M}^t_\varphi)\}) & = \\
\\
\mathcal{P}(\{w \in \mathit{IRun}(p(u_1X_1v_1)) \mid w \in \text{Ł}(\mathcal{M}'^s_\varphi)\}) & + \\
\sum_{(q,t)\in Q\times M_\varphi} \mathcal{P}(\{w\, q\varepsilon \in \mathit{Run}(p(u_1X_1v_1)) \mid w \in \text{Ł}(\mathcal{M}'^{s\longrightarrow t}_\varphi)\}) & \cdot \\
\mathcal{P}(\{w \in \mathit{IRun}(qX_2 \cdots X_n) \mid w_{\mathcal{B}} \in \text{Ł}(\mathcal{M}^t_\varphi)\}) & = \\
\\
[s, p(u_1X_1v_1) \bullet]_\varphi + \sum_{(q,t)\in Q\times M_\varphi} [s, p(u_1X_1v_1)q, t]_\varphi & \cdot \\
\mathcal{P}(\{w \in \mathit{IRun}(qX_2 \cdots X_n) \mid w_{\mathcal{B}} \in \text{Ł}(\mathcal{M}^t_\varphi)\}) &
\end{array}$$

Step 2: Let $\varphi = \mathcal{P}^{=x}\mathcal{B}$. We prove that $\mathcal{P}(\{w \in \mathit{IRun}(pX_1 \cdots X_n) \mid w_{\mathcal{B}} \in \text{Ł}(\mathcal{M}^s_\varphi)\}) = x$ if and only if $(p, s) \in u_1(\varphi)$. By induction on n.

First observe that $\mathcal{P}(\{w \in \mathit{IRun}(q\varepsilon) \mid w_{\mathcal{B}} \in \text{Ł}(\mathcal{M}^t_\varphi)\}) = 0$ for all $(q, t) \in Q \times M_\varphi$ and thus we may assume that $\mathcal{P}(\{w \in \mathit{IRun}(qX_2 \cdots X_n) \mid w_{\mathcal{B}} \in \text{Ł}(\mathcal{M}^t_\varphi)\}) = x$ if and only if $(q, t) \in v_1(\varphi)$. We prove the induction step. There are two cases.

- $\varphi = \mathcal{P}^{=1}\mathcal{B}$: Suppose that $(p, s) \in u_1(\varphi)$. By Definition 4.5 we have that $[s, p(u_1X_1v_1) \bullet]_\varphi + \sum_{(q,t)\in v_1(\varphi)} [s, p(u_1X_1v_1)q, t]_\varphi = 1$. Moreover, by induction hypothesis we have that $\mathcal{P}(\{w \in \mathit{IRun}(qX_2 \cdots X_n) \mid w_{\mathcal{B}} \in \text{Ł}(\mathcal{M}^t_\varphi)\}) = 1$ for all $(q, t) \in v_1(\varphi)$ and the rest follows from Step 1.

  Suppose that $\mathcal{P}(\{w \in \mathit{IRun}(pX_1 \cdots X_n) \mid w_{\mathcal{B}} \in \text{Ł}(\mathcal{M}^s_\varphi)\}) = 1$. By Step 1 we have that $[s, p(u_1X_1v_1) \bullet]_\varphi + \sum_{(q,t)\in Q\times M_\varphi} [s, p(u_1X_1v_1)q, t]_\varphi = 1$. Moreover, it follows from Step 1 that if $[s, p(u_1X_1v_1)q, t]_\varphi > 0$ then $\mathcal{P}(\{w \in \mathit{IRun}(qX_2 \cdots X_n) \mid w_{\mathcal{B}} \in \text{Ł}(\mathcal{M}^t_\varphi)\}) = 1$ which means that $(q, t) \in v_1(\varphi)$ by induction. But then $[s, p(u_1X_1v_1) \bullet]_\varphi + \sum_{(q,t)\in v_1(\varphi)} [s, p(u_1X_1v_1)q, t]_\varphi = 1$ and thus $(p, s) \in u_1(\varphi)$.

- $\varphi = \mathcal{P}^{=0}\mathcal{B}$: Suppose that $(p, s) \in u_1(\varphi)$. By Definition 4.5 we have that $[s, p(u_1X_1v_1) \bullet]_\varphi + \sum_{(q,t)\notin v_1(\varphi)} [s, p(u_1X_1v_1)q, t]_\varphi = 0$. Moreover, by induction hypothesis we have that $\mathcal{P}(\{w \in \mathit{IRun}(qX_2 \cdots X_n) \mid w_{\mathcal{B}} \in \text{Ł}(\mathcal{M}^t_\varphi)\}) = 0$ for all $(q, t) \in v_1(\varphi)$ and the rest follows from Step 1.

Suppose that $\mathcal{P}(\{w \in \mathit{IRun}(pX_1 \cdots X_n) \mid w_{\mathcal{B}} \in Ł(\mathcal{M}^s_\varphi)\}) = 0$. It follows from Step 1 that if $[s, p(u_1 X_1 v_1) q, t]_\varphi > 0$ then $\mathcal{P}(\{w \in \mathit{IRun}(qX_2 \cdots X_n) \mid w \in Ł(\mathcal{M}^t_\varphi)\}) = 0$ which means that $(q,t) \in \nu_1(\varphi)$ by induction hypothesis. But then

$$[s, p(u_1 X_1 v_1) \bullet]_\varphi + \sum_{(q,t) \notin \nu_1(\varphi)} [s, p(u_1 X_1 v_1) q, t]_\varphi = 0$$

and thus $(p,s) \in u_1(\varphi)$.

□

Lemma 4.7. *Let pX be a configuration of Δ. Then there exists an extension Δ^τ of Δ which is φ-consistent for each $\varphi \in \mathit{Acl}(\tau)$, and a configuration $p(uXv)$ which is φ-consistent in Δ^τ for each $\varphi \in \mathit{Acl}(\tau)$. Moreover, Δ^τ and $p(uXv)$ are effectively constructible is space which is* doubly exponential *in the size of τ (if τ is a PCTL formula, then the space complexity is only* singly exponential *in the size of τ) and* singly exponential *in the size of Δ.*

Proof. Let Δ' be an extension of Δ, $\varphi \in \mathit{Acl}(\tau)$, and $(uXv) \in \Gamma'$. Using the algorithm of Theorem 3.7, one can compute $u' \in St$ such that $(u'Xv)$ is φ-consistent in Δ' and for each $\psi \in \mathit{Acl}(\tau) \backslash \{\varphi\}$ we have that $u'(\psi) = u(\psi)$. The computation of u' takes space which is polynomial in $|\Delta'| + |\Sigma'_\varphi| + |\rho'_\varphi| + |\mathcal{F}_\varphi|$ and exponential in $|M_\varphi|$.

Using this fact and Lemma 4.6, we can easily design an algorithm which computes Δ^τ and $p(uXv)$ in the bottom-up fashion. Let us estimate the complexity of this algorithm. First, the size of an arbitrary extension Δ' of Δ is at most exponential in $|\Delta|$ and *doubly* exponential in $|\tau|$. If $\varphi \in \mathit{Acl}(\tau)$, then the size of Σ'_φ and ρ'_φ is at most exponential in $|\Delta|$ and *doubly* exponential in $|\tau|$. The size of M_φ and $\mathcal{F}_\varphi$ is at most exponential in $|\tau|$. The computation of Δ^τ and $p(uXv)$ needs at most $\mathcal{O}(|\tau| \cdot |\Delta^\tau|)$ applications of the step whose space costs have been just evaluated. Hence, the algorithm requires space which is singly exponential in $|\Delta|$ and doubly exponential in $|\tau|$.

In the case of PCTL, each of the $\mathcal{M}_\varphi$ automata is either the automaton corresponding to the $\mathcal{X}$ operator or the automaton corresponding to the $\mathcal{U}$ operator. Hence, there are in fact only two Muller automata whose size is fixed and hence constant. This means that the size of an arbitrary extension and the alphabet Σ'_φ is at most exponential in $|\Delta| \cdot |\tau|$ and using the same considerations as above we obtain that the algorithm requires space which is singly exponential in $|\Delta|$ and $|\tau|$. □

An immediate corollary to Lemma 4.6 and Lemma 4.7 is the following:

Theorem 4.8. *The model-checking problems for pPDA processes and the logics qPECTL* and qPCTL are in* ***2-EXPSPACE*** *and* ***EXPSPACE****, respectively.*

Finally, let us note that the construction presented in [Wal00] which shows **EXPTIME**-hardness of the model-checking problem for the logic CTL and PDA processes can be adapted so that it works for (non-probabilistic) BPA[3]. This idea carries over to the probabilistic case after some trivial modifications. Thus, we obtain the following (a full proof can be found in Appendix):

Theorem 4.9. *The model-checking problem for pBPA processes and the logic qPCTL is* ***EXPTIME****-hard.*

References

[ABIJ00] P.A. Abdulla, C. Baier, S.P. Iyer, and B. Jonsson. Reasoning about probabilistic channel systems. In *Proceedings of CONCUR 2000*, volume 1877 of *LNCS*, pages 320–330. Springer, 2000.

[AEY01] R. Alur, K. Etessami, and M. Yannakakis. Analysis of recursive state machines. In *Proceedings of CAV 2001*, volume 2102 of *LNCS*, pages 207–220. Springer, 2001.

[AR03] P.A. Abdulla and A. Rabinovich. Verification of probabilistic systems with faulty communication. In *Proceedings of FoSSaCS 2003*, volume 2620 of *LNCS*, pages 39–53. Springer, 2003.

[ASB+95] A. Aziz, V. Singhal, F. Balarin, R. Brayton, and A. Sangiovanni-Vincentelli. It usually works: The temporal logic of stochastic systems. In *Proceedings of CAV'95*, volume 939 of *LNCS*, pages 155–165. Springer, 1995.

[BE99] C. Baier and B. Engelen. Establishing qualitative properties for probabilistic lossy channel systems: an algorithmic approach. In *Proceedings of 5th International AMAST Workshop on Real-Time and Probabilistic Systems (ARTS'99)*, volume 1601 of *LNCS*, pages 34–52. Springer, 1999.

[BKS04] T. Brázdil, A. Kučera, and O. Stražovský. Deciding probabilistic bisimilarity over infinite-state probabilistic systems. In *Proceedings of CONCUR 2004*, volume 3170 of *LNCS*, pages 193–208. Springer, 2004.

[3] This observation is due to Mayr (Private communication, July 2004.)

[BS03] N. Bertrand and Ph. Schnoebelen. Model checking lossy channel systems is probably decidable. In *Proceedings of FoSSaCS 2003*, volume 2620 of *LNCS*, pages 120–135. Springer, 2003.

[Can88] J. Canny. Some algebraic and geometric computations in PSPACE. In *Proceedings of STOC'88*, pages 460–467. ACM Press, 1988.

[CSS03] J.M. Couvreur, N. Saheb, and G. Sutre. An optimal automata approach to LTL model checking of probabilistic systems. In *Proceedings of LPAR 2003*, volume 2850 of *LNCS*, pages 361–375. Springer, 2003.

[CY95] C. Courcoubetis and M. Yannakakis. The complexity of probabilistic verification. *JACM*, 42(4):857–907, 1995.

[EK99] J. Esparza and J. Knoop. An automata-theoretic approach to interprocedural data-flow analysis. In *Proceedings of FoSSaCS'99*, volume 1578 of *LNCS*, pages 14–30. Springer, 1999.

[EKM04] J. Esparza, A. Kučera, and R. Mayr. Model-checking probabilistic pushdown automata. In *Proceedings of LICS 2004*, pages 12–21. IEEE, 2004.

[EKS03] J. Esparza, A. Kučera, and S. Schwoon. Model-checking LTL with regular valuations for pushdown systems. *I&C*, 186(2):355–376, 2003.

[EY] K. Etessami and M. Yannakakis. Algorithmic verification of recursive probabilistic systems. Technical Report, School of Informatics, U. of Edinburgh, 2005.

[EY05] K. Etessami and M. Yannakakis. Recursive Markov chains, stochastic grammars, and monotone systems of non-linear equations. In *Proceedings of STACS'2005*, volume 3404 of *LNCS*. Springer, 2005. To Appear.

[Gri88] D. Grigoriev. Complexity of deciding Tarski algebra. *Journal of Symbolic Computation*, 5(1–2):65–108, 1988.

[HJ94] H. Hansson and B. Jonsson. A logic for reasoning about time and reliability. *Formal Aspects of Computing*, 6:512–535, 1994.

[HK97] M. Huth and M.Z. Kwiatkowska. Quantitative analysis and model checking. In *Proceedings of LICS'97*, pages 111–122. IEEE, 1997.

[HS84] S. Hart and M. Sharir. Probabilistic temporal logic for finite and bounded models. In *Proceedings of POPL'84*, pages 1–13. ACM Press, 1984.

[IN97] S.P. Iyer and M. Narasimha. Probabilistic lossy channel systems. In *Proceedings of TAPSOFT'97*, volume 1214 of *LNCS*, pages 667–681. Springer, 1997.

[Kwi03] M.Z. Kwiatkowska. Model checking for probability and time: from theory to practice. In *Proceedings of LICS 2003*, pages 351–360. IEEE, 2003.

[LS82] D. Lehman and S. Shelah. Reasoning with time and chance. *I&C*, 53:165–198, 1982.

[Min67] M.L. Minsky. *Computation: Finite and Infinite Machines*. Prentice-Hall, 1967.

[Rab03] A. Rabinovich. Quantitative analysis of probabilistic lossy channel systems. In *Proceedings of ICALP 2003*, volume 2719 of *LNCS*, pages 1008–1021. Springer, 2003.

[Var85] M. Vardi. Automatic verification of probabilistic concurrent finite-state programs. In *Proceedings of FOCS'85*, pages 327–338. IEEE, 1985.

[Wal00] I. Walukiewicz. Model checking CTL properties of pushdown systems. In *Proceedings of FST&TCS'2000*, volume 1974 of *LNCS*, pages 127–138. Springer, 2000.

5 Appendix

5.1 A formal proof of Theorem 4.1.

We construct a pPDA Δ which simulates the execution of $\mathcal{C}$. The stack alphabet of Δ is the set $\Gamma = \mathcal{T} \cup \{\#, A, B, A', B', \ell_1, \cdots, \ell_n\}$. Transition probabilities are not written explicitly, because we always assume uniform probability distribution over transitions with the same left-hand side. For example, by writing $pX \to qY$, $pX \to r\varepsilon$, and $pX \to pXY$, we implicitly define the probability of each transition to be $1/3$. This also means that our construction could be adjusted so that only transitions with probabilities 1 or $1/2$ were used.

The transition function of Δ is defined by the following rules that are split into four groups.

Group 1:

$$\begin{aligned}
\mathtt{init}\,\# &\to g_B\,\#\ell_1\# \\
g_B\,X &\to g_B\,BX \;\mid\; g_A\,X \\
g_A\,X &\to g_A\,AX \;\mid\; \mathtt{check}\,(\ell_i,\ell_j,r,d)\ell_j X \\
\mathtt{check}\,(\ell_i,\ell_j,r,d) &\to \mathtt{count}\,\alpha \;\mid\; \mathtt{state}\,\varepsilon \;\mid\; \mathtt{zero}\,\varepsilon \;\mid\; \beta \\
\mathtt{halt}\,X &\to \mathtt{halt}\,X
\end{aligned}$$

where $X \in \Gamma$, $(\ell_i, \ell_j, r, d) \in \mathcal{T}$, and

$$\alpha = \begin{cases} A & \text{if } r = 1 \text{ and } d = -1 \\ A' & \text{if } r = 1 \text{ and } d = 1 \\ B & \text{if } r = 2 \text{ and } d = -1 \\ B' & \text{if } r = 2 \text{ and } d = 1 \\ \varepsilon & \text{otherwise} \end{cases} \qquad \beta = \begin{cases} g_B\,\# & \text{if } j < n \\ \mathtt{halt}\,\varepsilon & \text{if } j = n \end{cases}$$

The rules in Group 1 are responsible for pushing configurations of $\mathcal{C}$ onto the stack. Each configuration is encoded by a sequence of symbols of the form $\ell_i A^{u_1} B^{u_2}$, and individual configurations are separated by #. Whenever a new configuration is pushed onto the stack, Δ enters a configuration of the form $\mathtt{check}\,(\ell_i, \ell_j, r, d)\ell_j A^{u_1} B^{u_2} \# \ell_k A^{v_1} B^{v_2} \# \gamma$. From this configuration, it is possible either to go on with guessing the next configuration (with probability $\frac{1}{4}$), or to proceed to some of the "testing" configurations. The tests ensure that the topmost configuration $\ell_j A^{u_1} B^{u_2}$ is the correct successor of the previous configuration $\ell_k A^{v_1} B^{v_2}$ if the instruction (ℓ_i, ℓ_j, r, d) is performed.

If Δ proceeds to the control state $\mathtt{count}$, then it is checked whether the counter values were updated correctly, i.e., whether $u_r = v_r + d$, and $u_s = v_s$ for the other counter s. This is somewhat tricky, because the value of the counter r stored in the topmost configuration is first adjusted by "undoing" the effect of the performed instruction (for example, if the counter represented by B's was incremented, the symbol B' is pushed. As we shall see, pushing an additional B' has the same effect as if we v_2 was incremented by one.) Then, the checking procedure verifies that the counter values stored in the two configurations are the same.

Group 2:

$$\begin{array}{rcl}
\mathtt{count}\, X & \to & \mathtt{new}_A\, X \mid \mathtt{old}_A\, X \mid \mathtt{new}_B\, X \mid \mathtt{old}_B\, X \\
\mathtt{new}_S\, S & \to & \mathtt{new}'_S\, \varepsilon \mid \mathtt{new}_S\, \varepsilon \\
\mathtt{new}_S\, \# & \to & \mathtt{new}_S\, \# \\
\mathtt{new}_S\, X & \to & \mathtt{new}_S\, \varepsilon \\
\mathtt{new}'_S\, X' & \to & \mathtt{new}'_S\, X'
\end{array}$$

where $X \in \Gamma \backslash \{A, B, \#\}$, $X' \in \Gamma$, and $S \in \{A, B\}$.

$$\begin{array}{rcl}
\mathtt{old}_S\, S' & \to & \mathtt{sink}\, \varepsilon \mid \mathtt{old}_S\, \varepsilon \\
\mathtt{old}_S\, \# & \to & \mathtt{old}'_S\, \varepsilon \\
\mathtt{old}_S\, X & \to & \mathtt{old}_S\, \varepsilon \\
\mathtt{old}'_S\, S & \to & \mathtt{sink}\, \varepsilon \mid \mathtt{old}'_S\, \varepsilon \\
\mathtt{old}'_S\, \# & \to & \mathtt{old}'_S\, \# \\
\mathtt{old}'_S\, X' & \to & \mathtt{old}'_S\, \varepsilon \\
\mathtt{sink}\, X'' & \to & \mathtt{sink}\, X''
\end{array}$$

where $X \in \Gamma \backslash \{\#, A', B'\}$, $X' \in \Gamma \backslash \{\#, A, B\}$, $X'' \in \Gamma$, and $S \in \{A, B\}$. The rules in Group 2 test the equality of counters in successive configurations.

Group 3:

$$\begin{array}{rcl}
\mathtt{state}\, \# & \to & \mathtt{state}'\, \varepsilon \\
\mathtt{state}\, X & \to & \mathtt{state}\, \varepsilon \\
\mathtt{state}'\, X' & \to & \mathtt{state}'\, X'
\end{array}$$

where $X \in \Gamma \backslash \{\#\}$ and $X' \in \Gamma$. The rules in Group 3 take care of the consistency of labels in successive configurations.

Group 4:

$$\begin{array}{rcl} \mathit{zero}\ \# & \rightarrow & \mathit{zero}'\ \varepsilon \\ \mathit{zero}\ X & \rightarrow & \mathit{zero}\ \varepsilon \\ \mathit{zero}'\ A & \rightarrow & \mathit{found}_A\ \varepsilon \ \mid\ \mathit{zero}'\ \varepsilon \\ \mathit{zero}'\ B & \rightarrow & \mathit{found}_B\ \varepsilon \ \mid\ \mathit{zero}'\ \varepsilon \\ \mathit{zero}'\ \# & \rightarrow & \mathit{zero}'\ \# \\ \mathit{zero}'\ X' & \rightarrow & \mathit{zero}'\ \varepsilon \end{array}$$

where $X \in \Gamma\backslash\{\#\}$ and $X' \in \Gamma\backslash\{A, B, \#\}$. The last group of rules is used to test the counters for zero.

The PCTL formula that is checked against the initial configuration $\mathit{init}\ \#$ of Δ is defined as follows:

$$\mathcal{P}^{>0}(\tau\, \mathcal{U}\ \mathit{halt})$$

where

$$\tau \equiv \bigwedge_{T=(\ell_i,\ell_j,r,d)\in\mathcal{T}} [\mathit{check}\ T \Rightarrow \mathcal{P}^{>0}\mathcal{X}(\mathit{count} \wedge \psi) \wedge \mathcal{P}^{>0}\mathcal{X}(\mathit{state} \wedge \phi_i) \wedge \mathcal{P}^{>0}\mathcal{X}(\mathit{zero} \wedge \chi_{r,d})]$$

where

$$\psi \equiv \mathcal{P}^{=\frac{1}{4}}(\mathcal{F}(\mathit{new}'_A \vee \mathit{old}'_A\ \#)) \wedge \mathcal{P}^{=\frac{1}{4}}(\mathcal{F}(\mathit{new}'_B \vee \mathit{old}'_B\ \#))$$

$$\phi_i \equiv \mathcal{P}^{>0}\mathcal{F}(\mathit{state}'\ \ell_i)$$

$$\chi_{r,d} \equiv \begin{cases} \mathcal{P}^{=0}\mathcal{F}(\mathit{found}_A) & \text{if } r = 1 \text{ and } d = 0 \\ \mathcal{P}^{=0}\mathcal{F}(\mathit{found}_B) & \text{if } r = 2 \text{ and } d = 0 \\ \mathit{true} & \text{otherwise} \end{cases}$$

Lemma 5.1. *Let σ be a configuration of Δ of the form*

$$\mathit{check}\ (\ell_i, \ell_j, r, d)\ell_j A^{u_1} B^{u_2} \# \ell_k A^{v_1} B^{v_2} \# \gamma$$

Then $\sigma \models \tau$ if and only if $k = i$, $u_r = v_r + d$, $u_s = v_s$ for $s \neq r$, and either $d \neq 0$ or $v_r = 0$.

Proof. By definition of τ we have that $\sigma \models \tau$ if and only if

$$\sigma \models \mathcal{P}^{>0}\mathcal{X}(\mathit{count} \wedge \psi) \wedge \mathcal{P}^{>0}\mathcal{X}(\mathit{state} \wedge \phi_i) \wedge \mathcal{P}^{>0}\mathcal{X}(\mathit{zero} \wedge \chi_{r,d})$$

By inspecting the rules in Group 3 and Group 4, it can easily be verified that $\sigma \models \mathcal{P}^{>0}\mathcal{X}(\mathit{state} \wedge \phi_i)$ iff $k = i$, and that $\sigma \models \mathcal{P}^{>0}\mathcal{X}(\mathit{zero} \wedge \chi_{r,d})$ iff either $d \neq 0$ or $v_r = 0$. Now let

$$\sigma' = \mathit{count}\ \alpha \ell_j A^{u_1} B^{u_2} \# \ell_k A^{v_1} B^{v_2} \# \gamma$$

We claim that $\sigma \models \mathcal{P}^{>0}\mathcal{X}(count \wedge \psi)$ if and only if $\sigma' \models \psi$. Let us consider, e.g., the case when $r = 1$ and $d = 1$, which means that $\alpha = A'$. Let us first evaluate the probability of the set of all runs initiated in σ' that satisfy the formula $\mathcal{F}(new'_A \vee old'_A \#)$ (we denote this set of runs by $Run(\sigma', \mathcal{F}(new'_A \vee old'_A \#))$; the same notation is used also for other configurations and path formulae). It follows easily from the rules in Group 2 that

$$\mathcal{P}(Run(\sigma', \mathcal{F}(new'_A))) = \frac{1}{4}(1 - \frac{1}{2^{u_1}})$$

Also observe that

$$\mathcal{P}(Run(\sigma', \mathcal{F}(old'_A \#))) = \frac{1}{4}\frac{1}{2}\frac{1}{2^{v_1}}$$

and that the two sets of runs are disjoint. Thus,

$$\mathcal{P}(Run(\sigma', \mathcal{F}(new'_A \vee old'_A \#))) = \frac{1}{4}(1 - \frac{1}{2^{u_1}}) + \frac{1}{4}\frac{1}{2}\frac{1}{2^{v_1}} = \frac{1}{4}(1 - \frac{1}{2^{u_1}} + \frac{1}{2^{v_1+1}})$$

which is equal to $\frac{1}{4}$ if and only if $u_1 = v_1 + 1$. The claim that $\sigma' \models \mathcal{P}^{=\frac{1}{4}}(\mathcal{F}(new'_B \vee old'_B \#))$ if and only if $u_2 = v_2$ is proven in a similar way. □

Lemma 5.2. $init \# \models \mathcal{P}^{>0}(\tau \,\mathcal{U}\, halt)$ *if and only if* $(\ell_1, 0, 0) \rightarrow^* (\ell_n, u_1, u_2)$ *for some values* u_1, u_2.

Proof. "⇒:" If $init \# \models \mathcal{P}^{>0}(\tau \,\mathcal{U}\, halt)$ then by inspecting the rules in Group 1 we obtain that there is a run $\pi = init \# \rightarrow g_B \#\alpha_0 \rightarrow^* check\ T_1\alpha_1 \rightarrow^* check\ T_k\alpha_k \rightarrow halt\ \alpha_k$ in $\mathcal{T}_\Delta$ such that $\alpha_0 = \ell_1\#$ and for each $1 \leq i \leq k$ it holds that $T_i \in \mathcal{T}$ and $\alpha_i = \ell_j A^{u_1} B^{u_2} \#\alpha_{i-1}$, where u_1, u_2 are some values and $1 \leq j \leq n$. Moreover, $check\ T_i\alpha_i \models \tau$ for each $1 \leq i \leq k$ and $T_k = (L_j, L_n, r, d)$. Thus, by Lemma 5.1 we obtain that the run π corresponds to a computation of $\mathcal{C}$ that reaches the label ℓ_n.

"⇐:" Suppose that there is a computation $c_1, \cdots, c_k$ of $\mathcal{C}$ such that $c_1 = (\ell_1, 0, 0)$ and $c_k = (\ell_n, u_1, u_2)$, where u_1, u_2 are some values, and c_{i+1} is obtained from c_i by performing a transition $T_i \in \mathcal{T}$ for each $1 \leq i < k$.

Now let us consider a run in $\mathcal{T}_\Delta$ of the form $\pi = init \# \rightarrow g_B \#\alpha_0 \rightarrow^* check\ T_1\alpha_1 \rightarrow^* check\ T_k\alpha_k \rightarrow halt\ \alpha_k$ where $\alpha_0 = \ell_1\#$ and $\alpha_i = \ell_j A^{u_1} B^{u_2} \#\alpha_{i-1}$ if $c_i = (\ell_j, u_1, u_2)$ for $1 \leq i \leq k$. It follows from Lemma 5.1 that $check\ T_i\alpha_i \models \tau$ for $1 \leq i \leq k$ and thus $\pi \models \tau \,\mathcal{U}\, halt$ since there are no other occurrences of the state $check$ in π. It follows that $init \# \models \mathcal{P}^{>0}(\tau \,\mathcal{U}\, halt)$. □

Theorem 4.1 is a simple corollary to Lemma 5.2. Observe that the nesting depth of $\mathcal{U}$ in the constructed formula $\mathcal{P}^{>0}(\tau \,\mathcal{U}\, halt)$ is indeed two, and that quantitative subformulae appear only in τ. It remains to explain how to modify the proof of Theorem 5.2

so that for all subformulae of the form $\mathcal{P}^{\sim\rho}\varphi$ we have that $\sim \in \{=, >\}$ and $\rho \in \{0, \frac{1}{2}, \frac{1}{4}, \frac{1}{8}\}$. We explain this just in the simplified setting of the example presented in Section 4. We have shown how to setup a pPDA system such that for all configurations of the form $pA^m\#A^n\#$ we have that $pA^m\#A^n\# \models \mathcal{P}^{=\frac{1}{2}}(\mathtt{tt}\,\mathcal{U}\,sA)$ iff $m = n$. Here we modify this example so that $pA^m\#A^n\# \models \neg\mathcal{P}^{>\frac{1}{4}}(\mathtt{tt}\,\mathcal{U}\,s_1A) \wedge \neg\mathcal{P}^{>\frac{1}{4}}(\mathtt{tt}\,\mathcal{U}\,s_2A)$ iff $m = n$. Let

$$pA \xrightarrow{1/2} p_1A, \quad pA \xrightarrow{1/2} p_2A,$$

$$p_1A \xrightarrow{1/2} q_1A, \quad q_1A \xrightarrow{1} q_1\varepsilon, \quad r_1A \xrightarrow{1/2} s_1A, \quad t_1A \xrightarrow{1/2} t_1\varepsilon, \quad s_1A \xrightarrow{1} s_1A,$$
$$p_1A \xrightarrow{1/2} t_1A, \quad q_1\# \xrightarrow{1} r_1\varepsilon, \quad r_1A \xrightarrow{1/2} r_1\varepsilon, \quad t_1A \xrightarrow{1/2} uA, \quad uA \xrightarrow{1} uA$$
$$t_1\# \xrightarrow{1} s_1A,$$

$$p_2A \xrightarrow{1/2} q_2A, \quad q_2A \xrightarrow{1} q_2\varepsilon, \quad t_2A \xrightarrow{1/2} s_2A, \quad r_2A \xrightarrow{1/2} r_2\varepsilon, \quad s_2A \xrightarrow{1} s_2A$$
$$p_2A \xrightarrow{1/2} t_2A, \quad q_2\# \xrightarrow{1} r_2\varepsilon, \quad t_2A \xrightarrow{1/2} t_2\varepsilon, \quad r_2A \xrightarrow{1/2} uA,$$
$$r_2\# \xrightarrow{1} s_2A,$$

By inspecting possible runs of $pA^m\#A^n\#$, one can easily confirm that the probability that a run of $pA^m\#A^n\#$ hits a configuration having s_1A as its head is exactly

$$\frac{1}{4}\cdot\frac{1}{2^m} + \frac{1}{2}\cdot(1-\frac{1}{2^n}) \quad = \quad \frac{1}{4} + \frac{1}{2^{m+1}} - \frac{1}{2^{n+1}}$$

Similarly, the probability that a run of $pA^m\#A^n\#$ hits a configuration having s_2A as its head is equal to

$$\frac{1}{4}\cdot\frac{1}{2^n} + \frac{1}{2}\cdot(1-\frac{1}{2^m}) \quad = \quad \frac{1}{4} + \frac{1}{2^{n+1}} - \frac{1}{2^{m+1}}$$

Thus, we obtain that $pA^m\#A^n\# \models \neg\mathcal{P}^{>\frac{1}{4}}(\mathtt{tt}\,\mathcal{U}\,s_1A) \wedge \neg\mathcal{P}^{>\frac{1}{4}}(\mathtt{tt}\,\mathcal{U}\,s_2A)$ iff $m = n$. Observe that the original formula where $\frac{1}{2}$ appeared as a quantitative constant was "split" into two formulae where $\frac{1}{4}$ is used as a quantitative constant. This is the reason why we also need the $\frac{1}{8}$ constant in Theorem 4.1; although this constant does not appear in the formula $\mathcal{P}^{>0}(\tau\,\mathcal{U}\,\mathtt{halt})$, it will appear in the modified formula. Also observe that the modification does not increase the nesting depth of the $\mathcal{U}$ operator.

5.2 A formal proof of Theorem 4.2.

Similarly as in the proof of Theorem 4.1, we construct a pBPA system Δ which simulates the execution of $\mathcal{C}$. Since there are no control states at our disposal, the checking phase is implemented differently. To see where the difference takes place, the rules are again structured into groups which have a similar purpose as before.

Group 1:

$$\begin{aligned}
S &\to G_{1,B}\#_1\ell_1\#_0\bot \\
G_{x,B} &\to G_{x,B}B_x \mid G_{x,A} \\
G_{x,A} &\to G_{x,A}A_x \mid (\ell_i,\ell_j,r,d)_x\,\ell_j \\
(\ell_i,\ell_j,r,d)_x &\to \mathtt{Count}\;\alpha \mid \mathtt{State} \mid \mathtt{Zero} \mid \beta \\
\bot &\to \bot
\end{aligned}$$

where $\alpha = \begin{cases} A_x & \text{if } r=1 \text{ and } d=-1 \\ A_{1-x} & \text{if } r=1 \text{ and } d=1 \\ B_x & \text{if } r=2 \text{ and } d=-1 \\ B_{1-x} & \text{if } r=2 \text{ and } d=1 \\ \varepsilon & \text{otherwise} \end{cases}$ $\beta = \begin{cases} G_{1-x,B}\#_{1-x} & \text{if } j<n \\ \mathtt{Halt} & \text{if } j=n \end{cases}$

$x \in \{0,1\}$, and $(\ell_i,\ell_j,r,d) \in \mathcal{T}$.

The rules in Group 1 generate a string of configurations of $\mathcal{C}$. The configurations are encoded as $\ell_i A_x^{u_1} B_x^{u_2}$ where the x alternates between 1 and 0 in order to distinguish between the leftmost configuration and its neighbouring configuration. The configurations are separated by $\#_x$ symbols. Whenever a new configuration is pushed onto the stack, then the leftmost symbol is $(\ell_i,\ell_j,r,d)_x$. Here it is possible (with probability $\frac{3}{4}$) to initiate some test. The tests ensure that the leftmost configuration is a correct successor of its neighbour according to the transition (ℓ_i,ℓ_j,r,d). The symbol α is again used to "undo" the effect of the instruction just performed. The symbol $\bot$ is added in order to make all runs infinite.

Group 2:

$$\begin{aligned}
\mathtt{Count} &\to N \mid O \\
N &\to \varepsilon \\
O &\to \varepsilon \\
A_x &\to A'_x \mid \varepsilon \\
B_x &\to B'_x \mid \varepsilon \\
A'_x &\to \varepsilon \\
B'_x &\to \varepsilon \\
\#_x &\to \varepsilon \\
L_j &\to \varepsilon
\end{aligned}$$

where $x \in \{0,1\}$ and $1 \le j \le n$.

Group 3:

$$\begin{aligned} \mathtt{State} &\to \varepsilon \\ \mathtt{Zero} &\to \varepsilon \\ \mathtt{Halt} &\to \varepsilon \end{aligned}$$

The PCTL$^+$ formula is defined as follows:

$$\mathcal{P}^{>0}((\tau_0 \wedge \tau_1)\, \mathcal{U}\, \mathtt{Halt})$$

where

$$\tau_x \equiv \bigwedge_{T=(\ell_i,\ell_j,r,d)\in\mathcal{T}} [T_x \Rightarrow \mathcal{P}^{>0}\mathcal{X}(\mathtt{Count} \wedge \psi_x) \wedge \mathcal{P}^{>0}\mathcal{X}(\mathtt{State} \wedge \phi_{x,i}) \wedge \mathcal{P}^{>0}\mathcal{X}(\mathtt{Zero} \wedge \chi_{x,r,d})]$$

$$\psi_x \equiv \mathcal{P}^{=\frac{1}{2}}\xi_x^A \wedge \mathcal{P}^{=\frac{1}{2}}\xi_x^B$$

$$\xi_x^A \equiv ((\neg O \wedge \neg \#_x)\, \mathcal{U}\, A'_x) \vee ((\neg N \wedge \neg A'_{1-x})\, \mathcal{U}\, \#_{1-x})$$

$$\xi_x^B \equiv ((\neg O \wedge \neg \#_x)\, \mathcal{U}\, B'_x) \vee ((\neg N \wedge \neg B'_{1-x})\, \mathcal{U}\, \#_{1-x})$$

$$\phi_{x,i} \equiv \mathcal{P}^{>0}((\neg \#_x)\, \mathcal{U}\, (\#_x \wedge \mathcal{P}^{>0}\mathcal{X}(L_i)))$$

$$\chi_{x,r,d} \equiv \begin{cases} \mathcal{P}^{=0}((\neg \#_{1-x})\, \mathcal{U}\, A_{1-x}) & \text{if } r = 1 \text{ and } d = 0 \\ \mathcal{P}^{=0}((\neg \#_{1-x})\, \mathcal{U}\, B_{1-x}) & \text{if } r = 2 \text{ and } d = 0 \\ \mathtt{true} & \text{otherwise} \end{cases}$$

Lemma 5.3. *Let* $\sigma = (\ell_i, \ell_j, r, d)_x\, \ell_j A_x^{u_1} B_x^{u_2} \#_x \ell_k A_{1-x}^{v_1} B_{1-x}^{v_2} \#_{1-x} \gamma$. *Then* $\sigma \models \tau_x$ *if and only if* $k = i$, $u_r = v_r + d$, $u_s = v_s$ *for* $s \neq r$, *and either* $d \neq 0$ *or* $v_r = 0$.

Proof. The proof is similar to the proof of Lemma 5.1. We show explicitly only that $\sigma' = \mathtt{Count}\ \alpha L_j A_x^{u_1} B_x^{u_2} \#_x L_k A_{1-x}^{v_1} B_{1-x}^{v_2} \#_{1-x} \gamma \models \psi_x$ if and only if $u_r = v_r + d$ and $u_s = v_s$ for $s \neq r$.

Let us consider, e.g., the case when $r = 1$ and $d = 1$, which means that $\alpha = A_{1-x}$. Let us evaluate the probability of $Run(\sigma', (\neg O \wedge \neg \#_x)\, \mathcal{U}\, A'_x)$. It is easy to see that

$$\mathcal{P}(Run(\sigma', (\neg O \wedge \neg \#_x)\, \mathcal{U}\, A'_x)) = \frac{1}{2}(1 - \frac{1}{2^{u_1}})$$

Now let us consider runs satisfying the formula $(\neg N \wedge \neg A'_{1-x})\, \mathcal{U}\, \#_{1-x}$. Such runs cannot pass through a configuration that has A'_{1-x} as its leftmost symbol (which plays the role of the state $\mathtt{sink}$ in the previous construction for pPDA). It is not hard to see that

$$\mathcal{P}(Run(\sigma', (\neg N \wedge \neg A'_{1-x})\, \mathcal{U}\, \#_{1-x})) = \frac{1}{2}\frac{1}{2}\frac{1}{2^{v_1}}$$

Moreover, the two sets of runs are disjoint since Count is surely rewritten either to O or toN. Thus,

$$\mathcal{P}(Run(\sigma', \xi_x^A)) = \frac{1}{2}(1 - \frac{1}{2^{u_1}}) + \frac{1}{2}\frac{1}{2}\frac{1}{2^{v_1}} = \frac{1}{2}(1 - \frac{1}{2^{u_1}} + \frac{1}{2^{v_1+1}})$$

that equals $\frac{1}{2}$ if and only if $u_1 = v_1 + 1$. The claim that $\mathcal{P}(Run(\sigma', \xi_x^B)) = \frac{1}{2}$ if and only if $u_2 = v_2$ is proven similarly. □

Lemma 5.4. $\mathtt{init} \# \models \mathcal{P}^{>0}((\tau_0 \wedge \tau_1)\, \mathcal{U}\, \mathtt{Halt})$ *if and only if* $(\ell_1, 0, 0) \rightarrow^* (\ell_n, u_1, u_2)$ *for some values* u_1, u_2.

Proof. Similar to the proof of Theorem 5.2. □

Again, Theorem 4.2 is a direct corollary to Lemma 5.4.

5.3 A full proof of Theorem 4.9

The proof is a modification of the construction presented in [Wal00]. An alternating Turing machine T is a tuple $(Q, \Gamma, \delta, q_I, q_A, q_R, \lambda)$, where λ is a function that partitions the set of states Q into existential and universal states, and q_I, q_A, q_R is the initial, accepting, and rejecting state, respectively. Let $T = (Q, \Gamma, \delta, q_I, q_A, q_R, \lambda)$ be an alternating Turing machine using n tape cells on input of size n. We construct a pBPA system Δ, a simple valuation ν, and a qPCTL formula φ such that for every configuration c of T there is configuration s of Δ such that $s \models^\nu \varphi$ iff T has an accepting computation from c. Since s is constructible in time which is polynomial in the size of c, we are (virtually) done.

We assume (w.l.o.g.) that the nondeterminism of T is limited so that the transition function of T assigns to each pair (state,letter) an ordered pair of moves (state,letter,direction) of T.

Let us define a pBPA system $\Delta = (\Gamma_\Delta, \delta_\Delta, Prob_\Delta)$ simulating computations of T on inputs of size n. The stack alphabet is $\Gamma_\Delta = Q \cup \Gamma \cup \{E, L, R\} \cup \{G, M, A, F\}$ where E, L, R are special letters. E stands for an arbitrary existential move. L and R stand for the left and the right element of a universal move, respectively. G, M, A, F are special control letters used for checking whether a computation is correct. The transition function δ_Δ contains the following rules (the probability function $Prob_\Delta$ is not significant, we only

need that it assigns a non-zero probability to each transition).

$$
\begin{array}{rcl@{\qquad\qquad}rcl}
G & \to & Mc'L & X & \to & \varepsilon \\
G & \to & Mc'E & q & \to & \varepsilon \\
G & \to & A & E & \to & \varepsilon \\
M & \to & G & R & \to & \varepsilon \\
A & \to & \varepsilon & L & \to & Mc'R \\
 & & & & & \\
M & \to & F & L & \to & F \\
F & \to & \varepsilon & & &
\end{array}
$$

where c' stand for a configuration of T (a string of length $n+1$), $X \in \Gamma$, and $q \in Q$. The simulation works as follows: It begins in the configuration Gc and starts guessing computation tree of T on c. Each of the configurations is guessed in n steps, because we need Δ to be polynomial in the size of T (hence, the rules $G \to Mc'L, G \to Mc'E, L \to Mc'R$ are in fact abbreviations for a family of rules that guess a new configuration symbol by symbol). Each time a new configuration is guessed, the run of Δ has to pass through a state with M on the top of the stack and the correctness is checked by the formula *Move*. When an accepting configuration is guessed (by putting A on the stack), and checked (by the formula *Accept*), all the symbols up to L are erased from the stack, and guessing of the right branch of the corresponding universal move is started. This continues until the stack becomes empty. The symbol F is used to detect that a run is going false in formula *Move*.

We define two formulas *Accept* and *Move* that check the correctness of the simulation performed by Δ.

$$\mathit{Accept} \equiv \mathcal{P}^{>0}\,(\neg(E \vee L \vee R)\,\mathcal{U}\,q_A)$$

The *Move* formula is slightly more complicated.

$$\mathit{Move} \equiv \mathcal{P}^{>0}\mathcal{X}\left(F \wedge \bigvee_{t\in\delta} \mathit{Trans}_t\right)$$

The Trans_t formulas are constructed from subformulas like

$$(\mathcal{P}^{>0}\mathcal{X})^{i+n+2}X \Rightarrow (\mathcal{P}^{>0}\mathcal{X})^{i}Y$$

in the standard way, using many next operators to check that the corresponding cells in the consecutive configurations are consistent with the transition function of T. This

part of the construction is omitted; the underlying principals are the same as in the construction presented in [Wal00].

The formula φ we are interested in looks as follows:

$$\varphi \equiv \mathcal{P}^{>0}\left((\neg \mathsf{F} \wedge (\mathsf{A} \Rightarrow \mathit{Accept}) \wedge (\mathsf{M} \Rightarrow \mathit{Move}))\,\mathcal{U}\,\varepsilon\right)$$

Now it is easy to check that $\mathsf{Gc} \models^{\nu} \varphi$ iff T has an accepting computation from c.