On the Decidability of Temporal Properties of Probabilistic Pushdown Automata

Tomáš Brázdil*, Antonín Kučera**, and Oldřich Stražovský

Faculty of Informatics, Masaryk University,
Botanická 68a, 60200 Brno, Czech Republic.
{brazdil,kucera,strazovsky}@fi.muni.cz

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., [22]) 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., [3, 16, 15]). 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), but they still retain a reasonable expressive power which is sufficient, e.g., for modelling some problems of interprocedural dataflow analysis [12]. 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.

* Supported by the Grant Agency of the Czech Republic, grant No. 201/03/1161.
** Supported by the Alexander von Humboldt Foundation and by the 1M National Research Centre "Institute for Theoretical Computer Science (ITI)".

[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.

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 [23, 19, 26, 18, 4, 10, 20, 11]. As for infinite-state systems, most works so far considered probabilistic lossy channel systems [21] which model asynchronous communication through unreliable channels [5, 1, 2, 6, 25]. The problem of deciding probabilistic bisimilarity over various classes of infinite-state probabilistic systems has recently been considered in [7]. Model-checking problems for pPDA and pBPA processes have been studied in [13]. In [13], 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 [16], 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 [13]. 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 [13] so that it works for Muller automata. Note that the considered problems are known to be **PSPACE**-hard even for finite-state systems [26]. 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 [27] which shows **EXPTIME**-hardness of the model-checking problem for (non-probabilistic) CTL and PDA. Due to space constraints, formal proofs are omitted. We refer to [8] for technical details.

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 1. *A* Büchi *automaton is a tuple $\mathcal{B} = (\Sigma, B, \varrho, b_I, Acc)$, where Σ is a finite* alphabet*, B is a finite set of* states*, $\varrho \subseteq B \times \Sigma \times B$ is a* transition relation *(we write $b \xrightarrow{a} b'$ instead of $(b, a, b') \in \varrho$), 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. *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 $\mathcal{P}(Run(w)) = \Pi_{i=1}^{|w|-1} x_i$ where $w(i{-}1) \xrightarrow{x_i} w(i)$ for every $1 \leq i < |w|$ (if $|w| = 1$, we put $\mathcal{P}(Run(w)) = 1$).

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

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).

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

Here a ranges over Ap, $\varrho \in [0,1]$, 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:

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

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 $\mathcal{T} = (S, \to, Prob)$ be a probabilistic transition system, and let $\nu : Ap \to 2^S$ 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.)

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

For PCTL, the semantics of path formulae is redefined to

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

The semantics of a PECTL* formula $\mathcal{P}^{\sim\varrho}\mathcal{B}$, 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 $w \in IRun$ 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 $Run(s, \mathcal{B}) = \{w \in IRun(s) \mid w_{\mathcal{B}} \in L(\mathcal{B})\}$. We stipulate that $s \models^\nu \mathcal{P}^{\sim\varrho}\mathcal{B}$ iff $\mathcal{P}(Run(s, \mathcal{B})) \sim \varrho$.

The *qualitative fragments* of PCTL, PCTL*, and PECTL*, denoted qPCTL, qPCTL*, and qPECTL*, resp., are obtained by restricting the allowed operator/number combinations in $\mathcal{P}^{\sim\varrho}\varphi$ and $\mathcal{P}^{\sim\varrho}\mathcal{B}$ 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 $\Delta = (Q, \Gamma, \delta, Prob)$ 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., [14]), and therefore they do not bring any extra expressive power.

Definition 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 $(s, \mathcal{C}_1, \mathcal{C}_2, \sim, \varrho)$, where $s \in S$, $\mathcal{C}_1, \mathcal{C}_2 \subseteq S$, $\sim \in \{\leq, <, \geq, >, =\}$, and $\varrho \in [0, 1]$. The question is if $\mathcal{P}(Run(s, \mathcal{C}_1 \,\mathcal{U}\, \mathcal{C}_2)) \sim \varrho$. In [13], 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 $(pX, \mathcal{C}_1, \mathcal{C}_2, \sim, \varrho)$, 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 $\mathcal{P}(\mathit{Run}(pX, \mathcal{C}_1 \,\mathcal{U}\, \mathcal{C}_2))$ is the first component in the tuple of non-negative real values which form the least solution of the system. Thus, the relation $\mathcal{P}(\mathit{Run}(pX, \mathcal{C}_1 \,\mathcal{U}\, \mathcal{C}_2)) \sim \varrho$ can effectively be expressed in $(\mathbb{R}, +, *, \leq)$ by constructing a formula Φ saying that a given vector $\boldsymbol{x}$ is the least solution of the system and $\boldsymbol{x}(1) \sim \varrho$. Since the quantifier alternation depth in the constructed formula is fixed, it was concluded in [13] that the random walk problem for pPDA and simple sets of configurations is in **EXPTIME** by applying the result of [17]. Later, it was observed in [16] 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 [13] just by observing that the existential (or universal) fragment of $(\mathbb{R}, +, *, \leq)$ is sufficient to decide whether $\mathcal{P}(\mathit{Run}(pX, \mathcal{C}_1 \,\mathcal{U}\, \mathcal{C}_2)) \sim \varrho$ when $\sim \in \{<, \leq\}$ (or $\sim \in \{>, \geq\}$, resp.). Since the existential and universal fragments of $(\mathbb{R}, +, *, \leq)$ are decidable in polynomial space [9], we obtain the following result which is used in our complexity estimations:

Lemma 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 [26]. Our proof is a generalization of the construction for *deterministic* Büchi automata presented in [13]. We show that this construction can be extended to (deterministic) Muller automata, which have the same expressive power as general Büchi automata.

Definition 5. *A* Muller automaton *is a tuple $\mathcal{M} = (\Sigma, M, \varrho, m_I, \mathcal{F})$, where Σ is a finite* alphabet*, M is a finite set of* states*, $\varrho\colon M \times \Sigma \to M$ is a (total)* transition function *(we write $m \xrightarrow{a} m'$ instead of $\varrho(m, a) = m'$), m_I is the* initial state*, and $\mathcal{F} \subseteq 2^M$ is a set of* accepting sets*.*

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

For the rest of this section, we fix a pPDA $\Delta = (Q, \Gamma, \delta, \mathit{Prob})$. 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 $\mathit{Run}(pX, \mathcal{M})$ be the set of all runs of $\mathit{IRun}(pX)$ that are accepted by $\mathcal{M}$. Our aim is to show that the problem if $\mathcal{P}(\mathit{Run}(pX, \mathcal{M})) \sim \varrho$ for given Δ, pX, $\mathcal{M}$, $\sim \in \{\leq, <, \geq, >\}$, and $\varrho \in [0, 1]$, is in **2-EXPTIME**. In the qualitative case, we derive the **EXPSPACE** upper bound.

Theorem 6. *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**.*

Corollary 7. *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 $\ell_1{:}inst_1, \cdots, \ell_n{:}inst_n$, 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. $c_r := c_r + 1;$ `goto` ℓ_j
Type II. if $c_r = 0$ `then goto` ℓ_j `else` $c_r := c_r - 1;$ `goto` ℓ_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 [24].

Our aim is to reduce the halting problem for Minsky machines to the PCTL model checking problem for pPDA. Since a full proof is somewhat technical, we give just an intuitive explanation and refer to [8] for missing details.

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: $(\ell_i, \ell_j, r, d)\ell_j A^{v_1} B^{v_2} \# \ldots$. 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 $pA^m\#A^n\# \models \mathcal{P}^{=\frac{1}{2}}(\mathtt{tt}\,\mathcal{U}\,p_{sA})$ 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 $\mathcal{P}^{\sim\varrho}\varphi$ which is employed in the proof we have that $\sim$ is $>$ and ϱ is a "simple" rational like $1/2$ or $1/4$. We refer to [8] for details.

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 [13]. 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).

Theorem 8. *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 $\mathcal{P}^{\sim\varrho}\varphi$ we have that $\sim$ is $>$.*

The proof of Theorem 8 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 8, 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 9. *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 $\mathcal{P}^{\sim\varrho}\varphi$ we have that $\sim$ is $>$.

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

Let $\varphi \equiv \mathcal{P}^{=x}\mathcal{B} \in Acl(\tau)$ 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 $\mathcal{M}_\varphi = (\Sigma_\varphi, M_\varphi, \varrho_\varphi, m^I_\varphi, \mathcal{F}_\varphi)$ 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 $St = \Pi_{\varphi\in Acl(\tau)} 2^{Q\times M_\varphi}$. For every $v \in St$, the projection of v onto a given $\varphi \in Acl(\tau)$ is denoted $v(\varphi)$. Note that $v(\varphi)$ is a set of pairs of the form (q, m), where $q \in Q$ and $m \in M_\varphi$.

Definition 10. *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 11. *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 $\mathcal{C}_\varphi = \{p(uXv) \mid pX \in f_\nu(a)$ 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 $\mathcal{M}'_\varphi = (\Sigma'_\varphi, M_\varphi, \varrho'_\varphi, m^I_\varphi, \mathcal{F}_\varphi)$, 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 13 below).

Definition 12. *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 $u(\varphi) = \{(p, s) \mid [s, p(uXv)\bullet]_\varphi + \sum_{(q,t)\in v(\varphi)} [s, p(uXv)q, t]_\varphi = 1\}$*
- *if $\varphi \equiv \mathcal{P}^{=0}\mathcal{B}$, then $u(\varphi) = \{(p, s) \mid [s, p(uXv)\bullet]_\varphi + \sum_{(q,t)\notin v(\varphi)} [s, p(uXv)q, t]_\varphi = 0\}$*

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 $1 \leq i \leq n$, and $v_i = u_{i+1}$ for every $1 \leq i < n$.

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 12 are effectively verifiable, because, e.g., the condition $[s, p(uXv)\bullet]_\varphi + \sum_{(q,t)\in v(\varphi)} [s, p(uXv)q, t]_\varphi = 1$ can effectively be translated into $(\mathbb{R}, +, *, \leq)$ using the construction of Theorem 6 and the results on random walks of [13] which were recalled in Section 2. We refer to [8] for details and complexity estimations.

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

Lemma 13. *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 $n \geq 1$) 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 $p(u_1X_1v_1) \in \mathcal{C}_\varphi$.*

Lemma 14. *Let pX be a configuration of Δ. Then there exists an extension Δ^τ of Δ which is φ-consistent for each $\varphi \in Acl(\tau)$, and a configuration $p(uXv)$ which is φ-consistent in Δ^τ for each $\varphi \in Acl(\tau)$. 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.*

An immediate corollary to Lemma 13 and Lemma 14 is the following:

Theorem 15. *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 [27] 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:

Theorem 16. *The model-checking problem for pBPA processes and the logic qPCTL is* ***EXPTIME****-hard.*

References

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

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

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

[4] 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*, vol. 939 of *LNCS*, pp. 155–165. Springer, 1995.

[5] 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)*, vol. 1601 of *LNCS*, pp. 34–52. Springer, 1999.

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

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

[8] T. Brázdil, A. Kučera, and O. Stražovský. On the decidability of temporal properties of probabilistic pushdown automata. Technical report FIMU-RS-2005-01, Faculty of Informatics, Masaryk University, 2005.

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

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

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

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

[12] J. Esparza and J. Knoop. An automata-theoretic approach to interprocedural data-flow analysis. In *Proceedings of FoSSaCS'99*, vol. 1578 of *LNCS*, pp. 14–30. Springer, 1999.
[13] J. Esparza, A. Kučera, and R. Mayr. Model-checking probabilistic pushdown automata. In *Proceedings of LICS 2004*, pp. 12–21. IEEE, 2004.
[14] J. Esparza, A. Kučera, and S. Schwoon. Model-checking LTL with regular valuations for pushdown systems. *I&C*, 186(2):355–376, 2003.
[15] K. Etessami and M. Yannakakis. Algorithmic verification of recursive probabilistic systems. Technical Report, School of Informatics, U. of Edinburgh, 2005.
[16] K. Etessami and M. Yannakakis. Recursive Markov chains, stochastic grammars, and monotone systems of non-linear equations. In *Proceedings of STACS'2005*, LNCS. Springer, 2005. To Appear.
[17] D. Grigoriev. Complexity of deciding Tarski algebra. *Journal of Symbolic Computation*, 5(1–2):65–108, 1988.
[18] H. Hansson and B. Jonsson. A logic for reasoning about time and reliability. *Formal Aspects of Computing*, 6:512–535, 1994.
[19] S. Hart and M. Sharir. Probabilistic temporal logic for finite and bounded models. In *Proceedings of POPL'84*, pp. 1–13. ACM Press, 1984.
[20] M. Huth and M.Z. Kwiatkowska. Quantitative analysis and model checking. In *Proceedings of LICS'97*, pp. 111–122. IEEE, 1997.
[21] S.P. Iyer and M. Narasimha. Probabilistic lossy channel systems. In *Proceedings of TAPSOFT'97*, vol. 1214 of *LNCS*, pp. 667–681. Springer, 1997.
[22] M.Z. Kwiatkowska. Model checking for probability and time: from theory to practice. In *Proceedings of LICS 2003*, pp. 351–360. IEEE, 2003.
[23] D. Lehman and S. Shelah. Reasoning with time and chance. *I&C*, 53:165–198, 1982.
[24] M.L. Minsky. *Computation: Finite and Infinite Machines*. Prentice-Hall, 1967.
[25] A. Rabinovich. Quantitative analysis of probabilistic lossy channel systems. In *Proceedings of ICALP 2003*, vol. 2719 of *LNCS*, pp. 1008–1021. Springer, 2003.
[26] M. Vardi. Automatic verification of probabilistic concurrent finite-state programs. In *Proceedings of FOCS'85*, pp. 327–338. IEEE, 1985.
[27] I. Walukiewicz. Model checking CTL properties of pushdown systems. In *Proceedings of FST&TCS'2000*, vol. 1974 of *LNCS*, pp. 127–138. Springer, 2000.