Analysis and Prediction of the Long-Run Behavior of Probabilistic Sequential Programs with Recursion (Extended Abstract)

Tomáš Brázdil[1,∗] Javier Esparza[2,†] Antonín Kučera[1,‡]

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

[2]Institute for Formal Methods in Computer Science, University of Stuttgart,
Universität str. 38, 70569 Stuttgart, Germany. esparza@informatik.uni-stuttgart.de

Abstract

We introduce a family of long-run average properties of Markov chains that are useful for purposes of performance and reliability analysis, and show that these properties can effectively be checked for a subclass of infinite-state Markov chains generated by probabilistic programs with recursive procedures. We also show how to predict these properties by analyzing finite prefixes of runs, and present an efficient prediction algorithm for the mentioned subclass of Markov chains.

1. Introduction

Probabilistic methods are widely used in the design, analysis, and verification of computer systems that exhibit some kind of "quantified uncertainty" such as coin-tossing in randomized algorithms, subsystem failures (caused, e.g., by communication errors or bit flips with an empirically evaluated probability), or underspecification in some components of the system [27]. The underlying semantic model are Markov chains or Markov decision processes, depending mainly on whether the systems under consideration are sequential or parallel. Properties of such systems can formally be specified as formulae of suitable temporal logics such as LTL, PCTL, or PCTL* [25]. In these logics, one can express properties like "the probability of termination is at least 98%", "the probability that each request will eventually be granted is 1", etc. Model-checking algorithms for these logics have been developed mainly for finite-state Markov chains and finite-state Markov decision processes [14, 31, 25, 13, 15]. This is certainly a limitation, because many implementations use unbounded data structures (counters, queues, stacks, etc.) that cannot always be faithfully abstracted into finite-state models. The question whether one can go beyond this limit has been rapidly gaining importance and attention in recent years. Positive results exist mainly for probabilistic lossy channel systems [6, 9, 26, 28, 2]. Examples of more generic results are [1, 29]. Very recently, probabilistic aspects of recursive sequential programs have also been taken into account [18, 11, 23, 21, 19, 22, 24]. In the non-probabilistic setting, the literature offers two natural models for such programs:

- *pushdown automata (PDA)*, see e.g. [17, 20, 32, 5], where the stack symbols correspond to individual procedures and their local data, and the global data is modeled in the finite-state control;
- *recursive state machines (RSM)*, see e.g. [4, 3], where the behavior of each procedure is specified by a finite-state automaton which can possibly invoke the computation of another automaton in a recursive fashion.

Since PDA and RSM are fully equivalent (in a well-defined sense) and there are linear-time translations between them, the results achieved for one model immediately apply to the other. A practical impact of these results can be documented by successful applications of software tools [7, 8].

Formal models for probabilistic recursive programs are obtained as probabilistic variants of PDA and RSM. The underlying semantics is given in terms of infinite-state Markov chains, and the two models are again equivalent with respect to this semantics. The existing results are described in greater detail in the following paragraph.

∗ Supported by the Czech Science Foundation, grant No. 201/03/1161.
† Partially supported by the EPSRC Grant "An Automata-theoretic Approach to Software Model Checking" and the DFG project "Algorithms for Software Model Checking".
‡ Supported by the Alexander von Humboldt Foundation and by the research center Institute for Theoretical Computer Science (ITI), project No. 1M0021620808.

In [18], it was shown that the generalized random walk problem for Markov chains generated by probabilistic PDA (pPDA for short) is decidable, and that the quantitative model-checking problem for deterministic Büchi specifications is also decidable. This study was continued in [11], where the result about deterministic Büchi automata was extended to deterministic Müller automata (and hence to all ω-regular properties). Moreover, it was shown that the model checking problem for the branching-time logic PCTL is already undecidable, while model-checking the qualitative fragment of the logic PECTL* is decidable. The complexity and other algorithmic aspects of the reachability problem for probabilistic RSM were studied in greater detail in [23]. In particular, it was shown that the qualitative reachability problem (i.e., the question whether the probability of reaching a given configuration from another given configuration is equal to 1) for one-exit probabilistic RSM is in **P**. The complexity of the model-checking problem for probabilistic RSM and ω-regular properties was studied in [21, 22]. In [19], it was shown how to compute the expected value and variance of the reward accumulated along a path between two configurations, and how to compute the average reward per transition for infinite paths.

Our Contribution. In this paper we focus on a different class of properties of probabilistic sequential programs which has not yet been considered in previous works. We are interested in *limit properties* of runs related to service cycles, and ways to efficiently *predict* them after performing (and observing) a bounded initial prefix of a run.

An important source of initial inspiration for this study was [16], where de Alfaro convincingly argues that conventional temporal logics cannot express important properties of the long-run average behavior of probabilistic systems. To get some intuition, consider a system which repeatedly services certain requests, like a www server, an answering machine, or a telephone switchboard. Typical performance or reliability questions like "What is the average time of servicing a request?" or "What is the probability that a request will be serviced within 3 seconds?" are not directly expressible in conventional temporal logics. In [16], each run of the system is assigned the average service time defined as $\lim_{n\to\infty}(\sum_{i=1}^{n} T(i))/n$, where $T(i)$ is the service time for the i^{th} request which appears along the run. Then, a special state predicate is introduced which holds in a given state iff the total probability of all runs where the average service time is bounded by a given constant is equal to 1. This predicate is then "plugged" into the syntax of temporal logics such as PCTL or PCTL*, and a model-checking algorithm for finite-state Markov decision processes is presented.

Various important reliability and performance properties cannot be deduced just from the average service time. Examples are the average deviation from the average service time, and the probability that a service takes longer than a given bound. To formulate such properties, we introduce a family of random variables that capture certain limit values of runs, and use these variables to define a family of *run-indicators*. A run-indicator classifies each run as "good" or "bad" according to these limit values, and one can thus formulate questions about the probability of good/bad behavior. For example, one can formally express questions like

- What is the probability that the average service time of a run is between 30 and 32 seconds?
- What is the probability of those runs where the average service time is between 30 and 32 seconds, and the average deviation from 31 seconds is at most 5 seconds?
- What is the probability of all runs satisfying the previous condition and the condition that the percentage of services longer than 37 seconds is at most 20%?

Actually, our treatment is generic in the sense that we use general reward functions to assign numeric values to individual services. These reward functions can also take negative values, and thus we can model arbitrary gains and costs (not only time). For pPDA, we restrict ourselves to non-negative reward functions whose values depend both on the current control state and the current stack content of a given PDA configuration (some of our results also work for reward functions that may take negative values; this is discussed in greater detail in Section 4). We show that the problem whether $\mathcal{P}(I{=}1) \sim \varrho$, where I is one of the introduced run-indicators, $\mathcal{P}(I{=}1)$ is the probability that I is satisfied, $\varrho \in [0,1]$ is a rational constant, and $\sim \in \{<,\leq,>,\geq,=\}$, is decidable. This allows to approximate $\mathcal{P}(I{=}1)$ by arbitrarily close rational lower and upper bounds (as we shall see, $\mathcal{P}(I{=}1)$ can be irrational).

Another issue addressed in this paper is the *prediction* of the aforementioned limit values. To the best of our knowledge, this problem has not yet been studied, and therefore we explain the underlying intuition in greater detail. In ergodic Markov chains, our limit random variables usually take just one value with probability one, regardless of the initial state of the run. For example, the average service time is the same for "almost all" runs, and hence it does not make much sense to predict it because its value is determined from the very beginning. (One can still ask "how fast" a run approaches this limit, but this is a different question not addressed in this paper.) However, in general Markov chains the average service time can take infinitely many values with a positive probability, and the probability that the average service time stays within given bounds changes along the execution of a run. Hence, one can ask whether it is possible to "predict" the future behavior just by inspecting a bounded prefix of a run. Of course, the answer is negative in general. However, we show that for the subclass of Markov chains that are definable by probabilistic PDA, such predictions *are* possible, even though these chains are infinite-state and non-ergodic. In fact, one can *efficiently* predict quite complicated run-indicators up to an arbitrarily small given error δ (the smaller δ we choose, the longer prefix of a run must be examined). We refer to Section 3 for precise definitions.

Finally, we study the decidability of the model-checking problem for temporal logics extended with state-predicates based on the limit features introduced in this paper. We prove that the model-checking problem remains decidable if we only use qualitative variants of these predicates, and derive an undecidability result for general predicates.

In this paper we rely on the results of [18, 23, 19]. Due to the lack of space, most results are stated without a proof. The proofs can be found in [10].

The paper is organized as follows. Section 2 contains preliminary definitions and some background information. In Section 3 we introduce a family of random variables that formally capture certain long-run average properties of Markov chains, and define the associated family of run-indicators. We also formalize the notion of prediction. In Section 4 we concentrate on probabilistic PDA and show how to compute and predict the properties introduced in Section 3. We also show how to handle the associated state predicates.

2. Preliminaries

In the paper we use $\mathbb{R}$ and $\mathbb{R}^+$ to denote the sets of real numbers and non-negative real numbers, respectively. We also use $\mathbb{R}_{\pm\infty}$ to denote $\mathbb{R} \cup \{-\infty, \infty\}$, and $\mathbb{R}^+_\infty$ to denote $\mathbb{R}^+ \cup \{\infty\}$. The symbols $-\infty, \infty$ are treated according to the standard conventions.

Markov chains. The underlying semantics of probabilistic sequential systems is defined in terms of discrete Markov chains.

Definition 2.1. *A (discrete)* Markov chain *is a triple* $M = (S, \to, Prob)$ *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* M *assigns its probability* $Prob(s \to t) \in (0,1]$ *so that for every* $s \in S$ *we have* $\sum_{s \to t} Prob(s \to t) = 1$.

In the rest of this paper we also write $s \xrightarrow{x} t$ instead of $Prob(s \to t) = x$. A *path* in M is a finite or infinite sequence $w = s_0, s_1, \cdots$ of states such that $s_i \to s_{i+1}$ for every i. The *length* of a given path w is the number of transitions in w. In particular, the length of an infinite path is ∞, and the length of a path s, where $s \in S$, is zero. We also use $w(i)$ to denote the state s_i of w (by writing $w(i) = s$ we implicitly impose the condition that the length of w is at least i). The prefix $s_0, \ldots, s_i$ of w is denoted by w^i. A *run* is an infinite path. The sets of all finite paths and all runs of M are denoted $FPath$ and Run, respectively. Similarly, the sets of all finite paths and runs that start with a given $w \in FPath$ are denoted $FPath(w)$ and $Run(w)$, respectively. In particular, $Run(s)$, where $s \in S$, is the set of all runs initiated in s.

In this paper we are interested in probabilities of certain events that are associated with runs. 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 $w \in FPath(s)$, and $\mathcal{P} : \mathcal{F} \to [0,1]$ is the unique probability function such that $\mathcal{P}(Run(w)) = \Pi_{i=0}^{m-1} x_i$ where $w = s_0, \cdots, s_m$ and $s_i \xrightarrow{x_i} s_{i+1}$ for every $0 \leq i < m$ (if $m = 0$, we put $\mathcal{P}(Run(w)) = 1$).

Probabilistic PDA. In this part we introduce probabilistic PDA, explain their basic features, and show how to overcome some of the fundamental difficulties of performing their quantitative analysis.

Definition 2.2. *A* probabilistic PDA (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* transition relation *such that whenever* $(p, X, q, \alpha) \in \delta$, *then* $|\alpha| \leq 2$, *and* $Prob$ *is a function which to each transition* $pX \to q\alpha$ *assigns a rational probability* $Prob(pX \to q\alpha) \in (0,1]$ *so that for all* $p \in Q$ *and* $X \in \Gamma$ *we have that* $\sum_{pX \to q\alpha} Prob(pX \to q\alpha) = 1$.

In the rest of this paper we adopt a more intuitive notation, writing $pX \to q\alpha$ instead of $(p, X, q, \alpha) \in \delta$, and $pX \xrightarrow{x} q\alpha$ instead of $Prob(pX \to q\alpha) = x$. The set $Q \times \Gamma^*$ of all configurations of Δ is denoted by $\mathcal{C}(\Delta)$. Given a configuration $pX\alpha$, we call pX the *head* and α the *tail* of $pX\alpha$.

To Δ we associate the Markov chain M_Δ where $\mathcal{C}(\Delta)$ is the set of states and the transitions are determined as follows:

- $p\varepsilon \xrightarrow{1} p\varepsilon$ for each $p \in Q$ (here ε denotes the empty stack);
- $pX\beta \xrightarrow{x} q\alpha\beta$ is a transition of M_Δ iff $pX \xrightarrow{x} q\alpha$ is a transition of Δ.

As a working example, we use a simple pPDA $\bar{\Delta}$ with two control states s, p, three stack symbols I, D, Z, and the following transitions:

$$sZ \xrightarrow{0.75} sZ, \quad sZ \xrightarrow{0.25} pIZ, \quad pI \xrightarrow{0.5} pID, \quad pI \xrightarrow{0.5} p\varepsilon,$$
$$pD \xrightarrow{0.5} pI, \quad pD \xrightarrow{0.5} pDD, \quad pZ \xrightarrow{1} pZ$$

The underlying Markov chain of $\bar{\Delta}$ is shown in Figure 1 (only the states reachable from sZ are drawn). Despite the simplicity of $\bar{\Delta}$, even basic questions about its behavior require a non-trivial attention. For example, one can ask what is the probability of reaching the "terminated" state pZ from the "initial" state sZ (formally, this probability is defined as $\mathcal{P}(\{w \in Run(sZ) \mid w(i) = pZ \text{ for some } i \in \mathbb{N}_0\})$). In this particular case, we can rely on standard results about one-dimensional random walks and answer that this probabil-

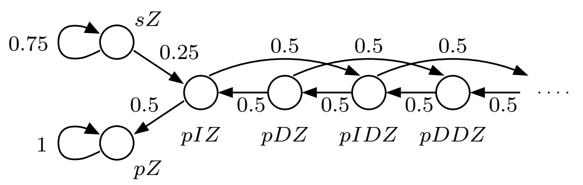

Figure 1. The Markov chain $M_{\bar{\Delta}}$

ity is equal to $(\sqrt{5}-1)/2$ (the "golden ratio"). This shows that the quantities of our interest can take irrational values.

Let $p\alpha$ and $q\beta$ be configurations of some pPDA Δ, and let $\mathcal{P}(p\alpha \rightarrow^* q\beta)$ be the probability of reaching $q\beta$ from $p\alpha$. In [18, 23], the reachability problem was solved by showing that $\mathcal{P}(p\alpha \rightarrow^* q\beta)$ is effectively expressible in $(\mathbb{R}, +, *, \leq)$. More precisely, there effectively exists a formula Φ of first order arithmetic of reals such that Φ has one free variable x and $\Phi[c/x]$ holds iff $c = \mathcal{P}(p\alpha \rightarrow^* q\beta)$. Since $(\mathbb{R}, +, *, \leq)$ is decidable [30], the problem whether $\mathcal{P}(p\alpha \rightarrow^* q\beta) \sim \varrho$, where $\sim \in \{<, \leq, =, >, \geq\}$ and ϱ is a rational constant, is decidable as well—it suffices to check whether the (closed) formula $\exists x.(\Phi \wedge x{\sim}\varrho)$ is valid or invalid. Hence, $\mathcal{P}(p\alpha \rightarrow^* q\beta)$ can also be effectively approximated—for an arbitrarily small $\delta > 0$ one can effectively compute rationals L, U such that $L \leq \mathcal{P}(p\alpha \rightarrow^* q\beta) \leq U$ and $U - L < \delta$. Since the formula Φ can be constructed so that the existential/universal quantifiers are not alternated in Φ and the size of Φ is polynomial in the size of $p\alpha$, $q\beta$, and Δ, one can apply the powerful result of [12] and conclude that the problem whether $\mathcal{P}(p\alpha \rightarrow^* q\beta) \sim \varrho$ is in **PSPACE**.

Observe that once a certain number (such as the probability of termination) is effectively expressible in $(\mathbb{R}, +, *, \leq)$ in the sense explained above, it can be used as a "known constant" in other first-order expressions which define other numbers. As long as these expressions contain just multiplication, addition, and inequality over reals, they can again be encoded into $(\mathbb{R}, +, *, \leq)$. Since nothing is actually *evaluated* during this process, there is no loss of precision and the newly expressed numbers enjoy essentially the same features as the "old" ones. In particular, they can be used as known constants when expressing other numbers, and their values can be effectively approximated. This approach has been used in [19] to express the conditional expected number of transitions needed to reach a configuration $q\varepsilon$ from pX under the condition that $q\varepsilon$ is indeed reached from pX. In this case, the "known" numbers are certain probabilities of the form $\mathcal{P}(rZ \rightarrow^* t\varepsilon)$.

The previous two paragraphs indicate how to overcome the problem that numerical features of our interest can take irrational values. Another fundamental difficulty is that Markov chains generated by pPDA are not necessarily ergodic. In fact, they are generally not strongly connected, and the number of strongly connected components can be infinite (even if M_Δ is strongly connected, all states can be transient or null). For finite-state systems, the problems considered in this paper could be solved relatively easily by employing known techniques for finite-state ergodic Markov chains. Our solution for pPDA is based on abstracting the Markov chain M_Δ into a finite-state Markov chain X_Δ so that certain properties of M_Δ can be determined by examining the corresponding properties of X_Δ. The definition and further discussion is postponed to Section 4.

3. Long-Run Properties of Markov Chains

In this section we introduce a family of long-run average properties of Markov chains. We show how to use these properties in performance analysis, and we also explain what is meant by a faithful and efficient prediction of these properties.

For the rest of this section, let us fix a Markov chain $M = (S, \rightarrow, \mathit{Prob})$ and an initial state $s_0 \in S$. We also fix a *reward function* $f : S \rightarrow \mathbb{R}$. The reward associated with a given state may correspond to, e.g., the time spent in the state, certain costs or gains collected by visiting the state (note that the reward can also be negative), or a one-bit marker specifying whether the state is "important" or not.

The request-service cycles are modeled as follows. Let $F \subseteq S$ be a subset of *triggers*. Let $w \in \mathit{Run}(s_0)$ be a run with infinitely many triggers $w(i_1), w(i_2), \ldots$, and let $w[j]$ denote the subword $w(i_{j-1}+1), \cdots, w(i_j)$ of w, where $i_0 = 0$. Hence, $w[j]$ is the subword of w consisting of all states in between the $j{-}1^{th}$ trigger (not included) and the j^{th} trigger (included). Intuitively, $w[j]$ corresponds to the j^{th} service. According to our definition, a new service starts immediately after finishing the previous service. (This is not a real restriction because the reward function can be set up so that the states visited before the actual start of the service are ignored, i.e., have zero reward.) Slightly abusing notation, we use $f(w[j])$ to denote the total reward accumulated in $w[j]$, i.e., $f(w[j]) = \sum_{k=i_{j-1}+1}^{i_j} f(w(k))$.

The properties of runs we are interested in here are formally defined as *indicators*. An indicator is a random variable $I : \mathit{Run}(s_0) \rightarrow \{1, 0\}$ which classifies the runs as "good" or "bad" according to some criterion. For example, the following simple indicator I_{inf} is obviously relevant in our setting:

$$I_{\mathit{inf}}(w) = \begin{cases} 1 & \text{if } w(i) \in F \text{ for infinitely many } i\text{'s;} \\ 0 & \text{otherwise.} \end{cases}$$

We are primarily interested in those runs w where $I_{\mathit{inf}}(w) = 1$, because only then the limit features introduced below make a good sense. The runs for which I_{inf} equals 0 are those where the service cycle is either eventually terminated, or the last service is never finished.

Since this can be seen as an error, $\mathcal{P}(I_{inf}=1)$ is an important quantitative information about the behavior of s_0. For example, the quantitative model-checking problem for linear-time properties definable via deterministic Büchi automata is obviously reducible to the problem of computing $\mathcal{P}(I_{inf}=1)$ in any class of models that is closed under synchronized product with a deterministic finite-state automaton (probabilistic PDA form such a class). The decidability of the model-checking problem for deterministic Büchi automata and pPDA has been shown in [18] by employing non-trivial methods. Hence, even computing $\mathcal{P}(I_{inf}=1)$ can be a difficult problem in general.

Before introducing other indicators, let us explain what is meant by "predictability" of an indicator.

Definition 3.1. *Let I be an indicator. We say that I is* well-predictable (over $Run(s_0)$) *if for each $\delta > 0$ there effectively exist $n \in \mathbb{N}$ and an indicator G^n such that $\mathcal{P}(G^n \neq I) \leq \delta$, and the value of $G^n(w)$ is effectively computable from the prefix of w of length n.*[1]

Hence, G^n efficiently "guesses" the value of I after seeing the first n states of a run, and the "quality" of that guess is measured by δ.

In general, indicators are rarely well-predictable. An important outcome of our work is that a large class of practically relevant indicators is well-predicable in the class of Markov chains generated by pPDA (at least, for those pPDA that satisfy a mild and effectively checkable condition formulated in Section 4).

Now we define other random variables and the associated indicators. Let $\kappa \in \mathbb{R}$, $\lambda, \gamma \in \mathbb{R}_{\pm\infty}$, and let $B(w[j], \lambda, \gamma)$ return either 1 or 0 depending on whether $\lambda \leq f(w[j]) \leq \gamma$ or not, respectively. We define the random variables

$$\begin{aligned} A(w) &= \lim_{n\to\infty} \frac{\sum_{j=1}^{n} f(w[j])}{n} \\ D[\kappa](w) &= \lim_{n\to\infty} \frac{\sum_{j=1}^{n} |f(w[j]) - \kappa|}{n} \\ R[\lambda,\gamma](w) &= \lim_{n\to\infty} \frac{\sum_{j=1}^{n} B(w[j],\lambda,\gamma)}{n} \end{aligned}$$

If the corresponding limit does not exist or $I_{inf}(w) = 0$, the above variables take the value $\perp$.

The variable A returns the average reward per service in a given run. The variable $D[\kappa]$ returns the average deviation of the reward per service from a given center κ in a given run. Finally, the variable $R[\lambda, \gamma]$ returns the percentage of services whose rewards are within the bounds λ, γ.

1 Due to the Markov property, the last state of the prefix contains a complete information that is relevant for predicting the future behavior; one cannot learn anything "fundamentally new" by inspecting the previous states in the prefix. However, this inspection can make the prediction more *efficient*, as we shall see in Section 4.

In general, $\mathcal{P}(I_{inf}=1 \wedge V=\perp)$, where V is one of the random variables introduced above, can be positive. However, as a byproduct of our results we obtain that this cannot happen for Markov chains generated by pPDA satisfying the condition of Section 4.

Let V be one of the above defined variables, and let $\ell, u \in \mathbb{R}_{\pm\infty}$. The indicator $I[V, \ell, u]$ is defined as follows:

$$I[V,\ell,u](w) = \begin{cases} 1 & \text{if } V(w) \neq \perp \text{ and } \ell \leq V(w) \leq u; \\ 0 & \text{otherwise.} \end{cases}$$

We also consider "Boolean combinations" of the indicators above (where 1 and 0 are interpreted as *true* and *false*, respectively). Thus, we obtain a family $\mathcal{I}$ consisting of I_{inf}, all $I[V, \ell, u]$, and their Boolean combinations. To show the relevance of Boolean combinations, let us formalize the properties mentioned in Section 1 (assume that the reward function corresponds to the time spent in a given state).

- $I[A, 30, 32]$ defines all runs where the average service time is between 30 and 32.
- $I[A, 30, 32] \wedge I[D[31], 0, 5]$ defines all runs where the average service time is between 30 and 32, and the average deviation of service time from 31 is at most 5.
- $I[A, 30, 32] \wedge I[D[31], 0, 5] \wedge I[R[37,\infty], 0, 0.2]$ defines all runs satisfying the previous condition and the condition that the percentage of services longer than 37 is at most 20%.

Let $I \in \mathcal{I}$. We study three basic algorithmic problems:

- Compute $\mathcal{P}(I=1)$. Since $\mathcal{P}(I=1)$ can be irrational for pPDA, we can only hope to solve this problem in the same way as the reachability problem was solved in [18, 23] (see Section 2). That is, the task is to show that $\mathcal{P}(I=1)$ is effectively expressible in $(\mathbb{R}, +, *, \leq)$. From this we obtain the decidability of the problem whether $\mathcal{P}(I=1) \sim \varrho$ for a given rational ϱ and $\sim \in \{<, \leq, >, \geq, =\}$. In particular, $\mathcal{P}(I=1)$ can be effectively approximated up to an arbitrarily small error (e.g., by a simple binary search).
- If I is well-predictable, design a suitable G^n. The indicator G^n should satisfy the "efficiency" requirements discussed after Definition 3.1.
- Since the predicate $\mathcal{P}(I=1) \sim \varrho$ is either valid or invalid in each state $s \in S$, it can be "plugged" into state-based temporal logics such as LTL, PCTL, or PCTL* in the style of [16] (the state predicate which has been introduced and studied in [16] corresponds to $\mathcal{P}(I[A, \ell, u]=1) = 1$). The question is whether there is a model-checking algorithm for these extended logics.

Note that the conditional probability $\mathcal{P}(I=1 \mid I_{inf}=1)$ (which is relevant in situations when $\mathcal{P}(I_{inf}=1)<1$) is expressible from the probabilities $\mathcal{P}(I=1 \wedge I_{inf}=1)$ and $\mathcal{P}(I_{inf}=1)$. Hence, if we manage to solve the three problems above, our results apply also to $\mathcal{P}(I=1 \mid I_{inf}=1)$.

4. Results for pPDA

In this section we examine and solve the three problems given at the end of Section 3 for pPDA and a general class of reward functions that take into account both the current control state and the current stack content. All these results work under a mild and effectively checkable condition, which is formulated and explained at the beginning of the next subsection.

For the rest of this section we fix a pPDA $\Delta = (Q, \Gamma, \delta, \mathit{Prob})$ and a subset $F \subseteq Q$ of control states. A configuration $p\alpha \in \mathcal{C}(\Delta)$ is a *trigger* iff $p \in F$. The notions introduced in Section 3 can now be applied to the chain M_Δ.

Since the problems formulated at the end of Section 3 are obviously undecidable for general reward functions, we restrict ourselves to the following subclass:

Definition 4.1. *A reward function $f : \mathcal{C}(\Delta) \to \mathbb{R}$ is* well-defined *if there are $g, h : Q \to \mathbb{R}^+$ and $c : \Gamma \to \mathbb{R}^+$ such that $f(p\alpha) = g(p) + h(p) \cdot (\sum_{Y \in \Gamma} c(Y) \cdot \#_Y(\alpha))$ for all $p\alpha \in \mathcal{C}(\Delta)$, where $\#_Y(\alpha)$ denotes the number of occurrences of Y in α. We say that f is* simple *(or* linear*) iff $h(p) = 0$ (or $h(p) = 1$, resp.) for all $p \in Q$.*

In the rest of this paper we use $c(\alpha)$ to denote $\sum_{Y \in \Gamma} c(Y) \cdot \#_Y(\alpha)$. Sometimes we abuse our notation by considering g and h as standalone simple reward functions.

In fact, for certain indicators in our family $\mathcal{I}$ (see Section 3) we can handle even more general reward functions where g and h can also take negative values. To simplify our presentation, we restrict ourselves only to those g and h that are non-negative. In the proof of Theorem 4.2 we explain in greater detail where and under what conditions we can deal with general g and h.

Simple reward functions can model gains and costs which do not depend on the history of activation records (i.e., the stack of procedure calls that have not terminated yet). A simple example is execution time—one can reasonably assume that the time spent in a given procedure for given input data does not depend on the current stack of activation records. On the other hand, if one is interested in e.g. memory consumptions, then the total amount of allocated memory in a given configuration does depend on the amount of memory allocated in the individual procedures stored in the stack, and here one can use linear reward functions. The reason why we also introduced the function h in Definition 4.1 is that in certain situations we wish not to "count" some configurations. For example, if we want to model an unbounded integer variable which is used in a given procedure, we might encode its value in unary by pushing a special symbol to the stack. Bounded changes to the variable (such as increment or decrement) can easily be implemented as single pPDA transitions. However, unbounded changes such as setting the variable back to 1 cannot be modeled as a single pPDA transition—the previously pushed symbols must be removed one by one. The artificially-added intermediate configurations can influence the properties of our interest, and hence the obtained results can become irrelevant. However, using h one can "switch off" the intermediate configurations so that they do not contribute to the accumulated reward.

As already mentioned in Section 2, Markov chains generated by pPDA are not necessarily ergodic. Nevertheless, questions about long-run average behavior are inherently related to concepts of ergodic chains (in particular, stationary distributions would be very useful in here). Fortunately, one can establish a surprisingly powerful link to this theory by abstracting the Markov chain M_Δ into another *finite-state* Markov chain X_Δ. The chain X_Δ has originally been introduced in [18]. Here we work with a slightly modified version of X_Δ which better suits our purposes, and present a collection of new results about X_Δ which are then used to solve the problems of our interest. The definition of X_Δ is given in the next subsection.

The Markov chain $\mathbf{X_\Delta}$. Let $pZ\alpha$ be a configuration of Δ, where $Z \in \Gamma$ and $\alpha \in \Gamma^*$. We say that a run $w \in \mathit{Run}(pZ\alpha)$ is *clean* if all configurations in w are of the form $q\beta\alpha$, where $\beta \in \Gamma^+$. In other words, α is never accessed in a clean run of $pZ\alpha$. In the rest of this section we study only properties of clean runs (we use $\mathit{Clean}(pZ\alpha)$ to denote the set of all clean runs of $\mathit{Run}(pZ\alpha)$ when defining certain conditional probabilities). This is no restriction, because we are actually interested in properties of runs from a given initial configuration q_0Z_0, which corresponds to the starting point of a given recursive sequential program. Since we can safely assume that Z_0 is a special bottom-of-the-stack marker which cannot be removed, all runs of $\mathit{Run}(q_0Z_0)$ are clean and our results apply.

For our purposes it suffices to consider clean runs initiated in configurations of the form pZ. Let $w = p_0\alpha_0, p_1\alpha_1 \cdots$ be a clean run of $\mathit{Run}(pZ)$ (i.e., $p_0\alpha_0 = pZ$). A configuration $p_i\alpha_i$ of w, where $i \geq 0$, is *minimal* if $|\alpha_i| \leq |\alpha_j|$ for all $j > i$. The *k-th minimum* of w, denoted $\min_k(w)$, is the k-th minimal configuration of w. The *index* of the k-th minimum, denoted $\mathit{ind}_k(w)$, is the i such that $w(i)$ is the k-th minimum of w. We say that $\min_k(w)$ is *increasing* if $k > 1$ and the stack length of $\min_k(w)$ is strictly larger than the one of $\min_{k-1}(w)$. Otherwise, $\min_k(w)$ is *non-increasing*.

Intuitively, the minimal configurations of a given run are exactly the positions where one can forget about the stack content below the top-of-the-stack symbol, because these symbols are never accessed in the future. This intuition is formally captured in our next definitions.

For all $p, q \in Q$ and $Z \in \Gamma$, we use $[pZq]$ to abbreviate $\mathcal{P}(pZ \to^* q\varepsilon)$, and $[pZ\uparrow]$ to abbreviate $1 - \sum_{r \in Q} [pZr]$. Hence, $[pZ\uparrow]$ is the probability that the stack never becomes

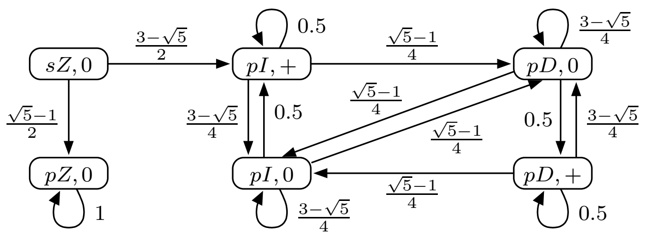

Figure 2. The Markov chain $X_{\bar{\Delta}}$

empty along a run of $Run(pZ)$. Equivalently, one can also say that $[pZ{\uparrow}]$ is the probability of $Clean(pZ)$.

For every configuration pZ and every $i \in \mathbb{N}$ we define a random variable X_i over $Run(pZ)$ as follows: if w is not clean, then $X_i = \bot$. Otherwise, $X_i(w) = (qY, m)$, where qY is the head of $\min_i(w)$, and m is either $+$ or 0 depending on whether $\min_i(w)$ is increasing or non-increasing, respectively. By adapting the proof technique of [18], one can easily show that for every $n \geq 2$ and all $(q_1Y_1, m_1), \cdots, (q_nY_n, m_n)$ such that $\mathcal{P}\big(\bigwedge_{i=1}^{n-1} X_i{=}(q_iY_i, m_i)\big) > 0$ we have that the probability

$$\mathcal{P}\big(X_n{=}(q_nY_n, m_n) \mid \bigwedge_{i=1}^{n-1} X_i{=}(q_iY_i, m_i)\big)$$

is equal either to $\displaystyle\sum_{q_{n-1}Y_{n-1} \xrightarrow{x} q_nY_nZ} \frac{x[q_nY_n{\uparrow}]}{[q_{n-1}Y_{n-1}{\uparrow}]}$ or to

$$\sum_{q_{n-1}Y_{n-1} \xrightarrow{x} rZY_n} \frac{x[rZq_n][q_nY_n{\uparrow}]}{[q_{n-1}Y_{n-1}{\uparrow}]} + \sum_{q_{n-1}Y_{n-1} \xrightarrow{x} q_nY_n} \frac{x[q_nY_n{\uparrow}]}{[q_{n-1}Y_{n-1}{\uparrow}]}$$

depending on whether m_n is equal to $+$ or to 0, respectively. In particular, observe that this probability is independent of the values of $X_1, \ldots, X_{n-2}$ and the value of n. Moreover, it is also independent of the initial configuration pZ. Hence, we can define a finite-state Markov chain X_Δ whose states are pairs of the form (qY, m), where $[qY{\uparrow}] > 0$, and the probability of $(qY, m) \to (q'Y', m')$ is given by the above term where $q_{n-1}Y_{n-1}$, q_nY_n, and m_n are substituted with qY, $q'Y'$, and m', respectively. More precisely, $(qY, m) \to (q'Y', m')$ is a transition in X_Δ iff the above term makes sense and produces a positive value which then defines the probability of this transition. Since this term contains only summation, multiplication, division, and probabilities of the form $[pXq]$ and $[pX{\uparrow}]$ which are known to be effectively expressible in $(\mathbb{R}, +, *, \leq)$ (see Section 2), we can conclude that the transition probabilities of X_Δ are also effectively expressible in $(\mathbb{R}, +, *, \leq)$.

As an example, consider again the pPDA $\bar{\Delta}$ defined in Section 2. The probability $[pIp]$ is equal to $(\sqrt{5} - 1)/2$, which means that $[pI{\uparrow}] = (3 - \sqrt{5})/2$. The Markov chain $X_{\bar{\Delta}}$ is depicted in Figure 2 (only the states reachable from $(sZ, 0)$ are drawn).

To each clean $w \in Run(pZ)$ we associate its *footprint* $X_1(w), X_2(w), \cdots$. Note that there can be clean runs whose footprints are *not* paths in X_Δ. For example, the run $sZ, sZ, sZ, \cdots$ of $Run(sZ)$ in $\bar{\Delta}$ has the footprint $(sZ, 0), (sZ, 0), (sZ, 0), \cdots$ which is not a path in $X_{\bar{\Delta}}$. Let $BSCC_\Delta$ be the set of all bottom strongly connected components of X_Δ (a BSCC is a nonempty subset C of states such that for all $s, t \in C$ we have that $s \to^* t$, and whenever $s \to^* u$, then $u \in C$). To each $C \in BSCC_\Delta$ we associate the set $Run(pZ, C)$ consisting of all $w \in Clean(pZ)$ such that the footprint of w is a path in X_Δ which hits the component C. We also define a random variable $Entry$ which for every $w \in Run(pZ)$ returns either $w(ind_j(w))$ where $j \in \mathbb{N}$ is the least number such that $X_j(w) \in C$ for some $C \in BSCC_\Delta$, or $\bot$ if there is no such j. In other words, if w is a clean run whose footprint hits a BSCC of X_Δ, then $Entry(w)$ is the configuration which "enters" this BSCC.

Note that since X_Δ has finitely many states, $\mathcal{P}(Run(pZ, C))$ is effectively expressible in $(\mathbb{R}, +, *, \leq)$ by employing standard methods for finite-state Markov chains (transition probabilities of X_Δ can be handled fully symbolically, there is no need to evaluate them). Moreover, it can easily be shown that

$$\sum_{C \in BSCC_\Delta} \mathcal{P}(Run(pZ, C) \mid Clean(pZ)) = 1$$

Consequently, $\mathcal{P}(Entry{=}\bot \mid Clean(pZ)) = 0$.

Solving the problems of Section 3 for pPDA. In this subsection we still work with the pPDA Δ which has been fixed at the beginning of Section 4. However, we need to adopt one additional assumption about Δ, which is crucial in almost all proofs:

"For all $p, q \in Q$ and $X \in \Gamma$, the conditional expected number of transitions needed to reach $q\varepsilon$ from pX, under the condition that $q\varepsilon$ is indeed reached from pX, is finite."

Formally, we define a random variable $Steps$ which to every $w \in Run(pX)$ assigns either the least j such that $w(j) = q\varepsilon$, or $\bot$ if there is no such j. Our assumption says that $E(Steps \mid Steps{\neq}\bot)$ is finite for all p, q, and X. Using the results of [19], one can effectively check in polynomial space whether this assumption is satisfied or not for a given pPDA. In terms of recursive sequential programs, the assumption corresponds to the requirement that if we restrict ourselves to terminating computations, then the expected termination time of each procedure is finite. From a practical point of view, this assumption is harmless because its violation indicates a severe design error anyway. From a theoretical point of view, this assumption allows to establish useful connections between the properties of M_Δ and X_Δ, as we shall see in the forthcoming theorems.

To simplify our notation, for the rest of this section we fix a well-defined reward function f and a distinguished *initial configuration* q_0Z_0 of Δ such that the symbol Z_0 cannot be removed from the stack (this assumption is not restrictive because one can always add a special bottom-of-the stack symbol without influencing the behavior of a given pPDA). Hence, all runs of q_0Z_0 are clean. Some of our results are formulated for configurations of the form pZ (which means that they hold generally), and some of them are formulated for the distinguished initial configuration q_0Z_0.

We start by presenting a crucial result which says that the (in)validity of all indicators in our family $\mathcal{I}$ for a given $w \in Run(pZ)$ is essentially determined only by the BSCC of X_Δ hit by w, and by the stack content in the configuration which enters this component. To formulate this precisely, we need to introduce another indicator $Hit[L]$, where $L \subseteq \Gamma^*$ is a regular language:

$$Hit[L](w) = \begin{cases} 1 & \text{if } Entry(w) = pX\beta \text{ and } \beta \in L; \\ 0 & \text{otherwise.} \end{cases}$$

Theorem 4.2. *Let $I \in \mathcal{I}$ be an indicator and pZ a configuration of Δ. For every $C \in BSCC_\Delta$ there effectively exists a regular language $L_C \subseteq \Gamma^*$ such that $\mathcal{P}(I = Hit[L_C] \mid Run(pZ, C)) = 1$. Moreover, if the considered reward function f is simple, then L_C equals either Γ^* or $\emptyset$.*

Proof sketch. Obviously, it suffices to consider indicators of the form I_{inf} and $I[V, \ell, u]$ (Boolean connectives are then no problem, because these can be implemented just by performing an appropriate operation on the constructed regular languages).

First, let us consider the indicator I_{inf}. It can be proved (by adapting the methods of [19]) that the indicator I_{inf} returns the same value for almost all runs of $Run(pZ, C)$, and that this value is expressible. Hence, the language L_C is equal either to Γ^* or to $\emptyset$.

Let us consider an indicator of the form $I[V, \ell, u]$. The argument can be split into two parts. First, we consider the special case when the Markov chain X_Δ is ergodic. We show that then there is a constant $\theta \in \mathbb{R}^+_\infty$ expressible in $(\mathbb{R}, +, *, \leq)$ such that for almost all clean runs w of $Run(pZ)$ we have that $V(w) = \theta$. This is the most involved construction in this paper. It requires specific techniques for each of the three variables A, $D[\kappa]$, and $R[\lambda, \gamma]$. In the case of A we can also handle well-defined reward functions f where the underlying g and h may take *negative* values (under some additional assumptions, which are satisfied, e.g., for all simple reward functions). However, we did not manage to extend this result to $D[\kappa]$ and $R[\lambda, \gamma]$, and therefore we adopted the simplified setting in Definition 4.1.

Then, we consider the general case. Since each BSCC C of X_Δ is an ergodic Markov chain and since the value of V does not depend on a finite prefix of a run, we have that V returns the same value $\xi(\beta, C)$ for almost all runs whose footprints enter the component C with a given stack contents β. Moreover, $\xi(\beta, C)$ is expressible as follows. We define a new reward function f' by putting $f'(p\alpha) = (g(p) + h(p) \cdot c(\beta)) + h(p) \cdot c(\alpha)$. Hence, $f'(p\alpha) = f(p\alpha\beta)$. Now we choose an arbitrary pX such that $(pX, m) \in C$ for some m. It is easy to see that for almost all clean $w \in Run(pX)$ we have that $V(w) = \xi(\beta, C)$, where $V(w)$ is computed with respect to the new reward function f'. Now it suffices to apply the results of the previous paragraph.

Let us define L_C to be the set of all $\beta \in \Gamma^*$ such that $\ell \leq \xi(C, \beta) \leq u$. It follows from our discussion that $\mathcal{P}(I = Hit[L_C] \mid Run(q_0Z_0, C)) = 1$. We prove that the language L_C is regular. The proof is based on the following observation: One can show that there effectively exists $n \in \mathbb{N}$ such that for all $\beta, \gamma \in \Gamma^*$ where $c(\beta) > n$ and $c(\gamma) > n$ we have that $\beta \in L_C$ iff $\gamma \in L_C$. The proof is obtained by considering the variables A, $D[\kappa]$, and $R[\lambda, \gamma]$ separately, the argument is not generic.

Let $\Gamma_{c>0} = \{Z \in \Gamma \mid c(Z) > 0\}$ be the set of c-*important* symbols. Observe that for each $\beta \in \Gamma^*$, the value of $c(\beta)$ depends only on the subsequence of c-important symbols contained in β, which we call the c-*span* of β. For a given $\gamma \in \Gamma^*_{c>0}$, let $L(\gamma)$ be the set of all $\beta \in \Gamma^*$ whose c-span is γ. Obviously, each $L(\gamma)$ is a regular language. Moreover, we either have that $L(\gamma) \subseteq L_C$, or $L(\gamma) \cap L_C = \emptyset$. Due to the observation formulated in the previous paragraph, we see that there is an effectively computable constant n' such that the union of all $L(\gamma')$, where the length of γ' is larger than n', either forms a subset of L_C, or is disjoint with L_C. Now it is easy to see that L_C is effectively regular. □

Theorem 4.3. *Let $L \subseteq \Gamma^*$ be a regular language, pZ a configuration of Δ, and $C \in BSCC_\Delta$. Then $\mathcal{P}(Hit[L]{=}1 \mid Run(pZ, C))$ is effectively expressible in $(\mathbb{R}, +, *, \leq)$.*

Proof sketch. We construct another pPDA Δ' which has the same stack alphabet as Δ and a special control state $succ$ such that $\mathcal{P}(Hit[L]{=}1 \mid Run(pZ, C))$ is equal to the conditional probability of reaching a configuration of the form $succ\ \alpha$ in Δ', where $\alpha \in L$, under the condition that a configuration with the control state $succ$ is reached. Since L is regular, this conditional probability can be expressed using the results about random walks presented in [18]. □

As a corollary to Theorem 4.2 and Theorem 4.3 we obtain the following (where q_0Z_0 plays the role of initial configuration):

Theorem 4.4. *Let $I \in \mathcal{I}$ be an indicator. The probability $\mathcal{P}(I{=}1)$ is effectively expressible in $(\mathbb{R}, +, *, \leq)$. Moreover, if the considered reward function f is simple, then the size of the resulting formula is polynomial in the size of Δ, and the alternation depth of quantifiers is fixed.*

Proof sketch. Due to Theorem 4.2, Theorem 4.3, and the fact that all runs of $Run(q_0Z_0)$ are clean we obtain that $\mathcal{P}(I{=}1)$ equals

$$\sum_{C \in BSCC_\Delta} \mathcal{P}(Hit[L_C]{=}1 \mid Run(q_0Z_0, C)) \cdot \mathcal{P}(Run(q_0Z_0, C))$$

where the probabilities $\mathcal{P}(\mathit{Hit}[L_C]{=}1 \mid \mathit{Run}(q_0Z_0, C))$ and $\mathcal{P}(\mathit{Run}(q_0Z_0, C))$ are expressible in $(\mathbb{R}, +, *, \leq)$. Hence, $\mathcal{P}(I{=}1)$ is also expressible. The result about simple reward functions follows by a detailed analysis of the constructions employed in Theorem 4.2 and Theorem 4.3. $\square$

Consequently, the problem whether $\mathcal{P}(I{=}1) \sim \varrho$, where ϱ a rational constant and $\sim \in \{<, \leq, >, \geq, =\}$, is decidable. Moreover, for simple reward functions we obtain an **EXPTIME** upper bound. Theorem 4.2 can also be used to prove the following:

Theorem 4.5. *Each $I \in \mathcal{I}$ is well-predictable (over $\mathit{Run}(q_0Z_0)$).*

Proof sketch. Let $\delta > 0$. Due to Theorem 4.2, it suffices to compute a sufficiently large n satisfying the following property: the probability of all $w \in \mathit{Run}(q_0Z_0)$ such that the position of $\mathit{Entry}(w)$ in w is beyond the prefix of length n, is bounded by δ. The value of $G^n(w)$ is then defined as follows: we take the first n configurations of w and identify the "developing minimal configurations", i.e., those configurations which become minimal configurations of w under the assumption that the stack length in all configurations $w(n), w(n{+}1), \ldots$ is not smaller than in $w(n{-}1)$. Thus, we also construct the "developing footprint" of the run. Then we simply check whether this developing footprint hits a BSCC of X_Δ. If not, $G^n(w)$ returns 0. Otherwise, we identify the Entry configuration $pX\beta$ and check whether $\beta \in L_C$, where C is the corresponding BSCC. If $\beta \in L_C$, then $G^n(w) = 1$. Otherwise, $G^n(w) = 0$. $\square$

Observe that the algorithm for computing $G^n(w)$ for given n and w is rather efficient, because the developing minima are identified just by comparing the stack length in configurations $w(0), \ldots, w(n{-}1)$. Of course, we also need to compute the transitions of X_Δ (which can be done in polynomial space in the size of Δ), but this expensive computation is performed just once and can be done before starting the on-line analysis of a run initiated in q_0Z_0.

Model-checking temporal logics with state predicates. Let M be a Markov chain, $I \in \mathcal{I}$ an indicator, and f a reward function. For every $\sim \in \{<, \leq, >, \geq, =\}$ and every rational constant ϱ we define a *state predicate* $\mathcal{P}^{\sim\varrho}(I{=}1)$ as follows: a state s of M satisfies $\mathcal{P}^{\sim\varrho}(I{=}1)$ iff $\mathcal{P}(I{=}1) \sim \varrho$ (here $\mathcal{P}(I{=}1)$ is considered in the probabilistic space over $\mathit{Run}(s)$). State predicates can be plugged into state-based temporal logics (such as LTL, PCTL, or PCTL*) in the style of [16], and thus one can combine the expressive power of state predicates with temporal operators. As we already mentioned in Section 1, the model-checking problem with pPDA is decidable for ω-regular properties [18, 11, 21] and the qualitative fragment of PECTL* [11].

First we show that if these logics are extended with predicates of the form $\mathcal{P}^{=1/2}(I{=}1)$, then even model checking the simple formula $\Diamond^{>0}(\mathit{check} \wedge \mathcal{P}^{=1/2}(I{=}1))$ becomes undecidable (the formula says "there is a reachable state satisfying the predicates check and $\mathcal{P}^{=1/2}(I{=}1)$"). Observe that the undecidability result does *not* hold for all $I \in \mathcal{I}$, because some of these indicators are trivial (for example, $I[A, 10, 11] \wedge I[A, 12, 13]$ is equivalent to *false*, which makes the considered temporal formula unsatisfiable and hence trivially decidable). We say that $I \in \mathcal{I}$ is *nontrivial* if there is a pPDA Δ_I and configurations tY, fY such that $tY \models \mathcal{P}^{=1}(I{=}1)$ and $fY \models \mathcal{P}^{=0}(I{=}1)$. For technical convenience, we also require that $[tY\!\uparrow] = [fY\!\uparrow] = 1$.

Theorem 4.6. *Let $I \in \mathcal{I}$ be a nontrivial indicator. Then the model-checking problem for pPDA and the formula $\Diamond^{>0}(\mathit{check} \wedge \mathcal{P}^{=1/2}(I{=}1))$ is undecidable.*

Proof sketch. We reduce (a slightly modified version of) the PCP problem: An instance are two sequences $x_1, \ldots, x_n$ and $y_1, \ldots, y_n$ of words over the alphabet $\Sigma = \{a, b, \bullet\}$ such that $|x_i| = |y_i|$ for each $1 \leq i \leq k$. The question is whether there is a finite sequence $i_1, \cdots, i_k$ of indexes such that $x_{i_1} \cdots x_{i_k}$ and $y_{i_1} \cdots y_{i_k}$ are the same words after erasing all occurrences of "$\bullet$".

For a given instance of PCP, we construct a pPDA Δ and its configuration gZ such that $gZ \models \Diamond^{>0}(\mathit{check} \wedge \mathcal{P}^{=1/2}(I{=}1))$ iff the PCP instance has a solution. The pPDA Δ is obtained by extending the pPDA Δ_I which witnesses the non-triviality of I. The considered reward function does not count the newly-added stack symbols, which means that $tY\beta \models \mathcal{P}^{=1}(I{=}1)$ and $fY\beta \models \mathcal{P}^{=0}(I{=}1)$ for an arbitrary sequence β of the newly-added stack symbols (here we use the assumption that $[tY\!\uparrow] = [fY\!\uparrow] = 1$).

From the (new) initial configuration gZ, the automaton Δ tries to "guess" a solution to our instance of PCP by storing pairs of words (x_i, y_i) successively to the stack. Since x_i and y_i have the same length, this is implemented by pushing pairs of letters from Σ. For example, if $x_i = aab$ and $y_i = ba\bullet$, then the pair (x_i, y_i) is stored as a sequence of three stack symbols $(a, b), (a, a), (b, \bullet)$. After storing a chosen pair of words, the automaton can either go on with guessing another pair of words, or enter a *checking* configuration. This is done by changing the control state from g to c. The predicate check is satisfied in exactly all checking configurations. The transition probabilities do not matter here, they can have arbitrarily non-zero values. The crucial part of the construction is the next phase where we verify that the guess was correct, i.e., that the words stored in the first and the second component of stack symbols are the same (when "$\bullet$" is disregarded). For this we use the following transitions (since the probability distribution is always uniform, we do not write the transition probabilities explicitly; the symbol "|" separates alternatives):

$$\begin{array}{lll}
cX \to vX \mid \hat{v}X, & v(a,z) \to tY \mid v\varepsilon, & \hat{v}(z,a) \to fY \mid \hat{v}\varepsilon, \\
 & v(b,z) \to fY \mid v\varepsilon, & \hat{v}(z,b) \to tY \mid \hat{v}\varepsilon, \\
 & v(\bullet,z) \to v\varepsilon, & \hat{v}(z,\bullet) \to \hat{v}\varepsilon, \\
 & vZ \to tY \mid fY, & \hat{v}Z \to tY \mid fY
\end{array}$$

Here z ranges over Σ, and X ranges over the stack alphabet. We claim that the checking configuration satisfies the predicate $\mathcal{P}^{=1/2}(I{=}1)$ iff the previous guess was correct. To see this, realize that the checking configuration satisfies $\mathcal{P}^{=1/2}(I{=}1)$ iff the probability of reaching a configuration having tY as its head is exactly $\frac{1}{2}$. To reveal the subtlety of the construction, let us evaluate this probability for, e.g., a configuration $g(a,a)(a,\bullet)(\bullet,a)(b,b)Z$. By inspecting the above rules, one can easily confirm that the probability is equal to

$$\frac{1}{2}\left(\left(1\cdot\frac{1}{2}+1\cdot\frac{1}{2^2}+0\cdot\frac{1}{2^3}+1\cdot\frac{1}{2^4}\right)+\left(0\cdot\frac{1}{2}+0\cdot\frac{1}{2^2}+1\cdot\frac{1}{2^3}+1\cdot\frac{1}{2^4}\right)\right)$$

Hence, this probability is equal to $\frac{1}{2}(0.1101+0.0011)$. The binary numbers 0.1101 and 0.0011 are "complementary" and their sum is equal to 1. It is easy to verify that this "complementarity" breaks down iff the words stored in the first and the second component of stack symbols are *not* the same, in which case the probability is different from $\frac{1}{2}$. $\square$

Now we prove that if we only allow qualitative predicates of the form $\mathcal{P}^{=1}(I{=}1)$, the situation becomes different:

Theorem 4.7. *The model-checking problem for pPDA and ω-regular properties as well as qualitative PECTL* formulae extended with qualitative state predicates of the form $\mathcal{P}^{=1}(I{=}1)$, where $I \in \mathcal{I}$, is decidable.*

In particular, Theorem 4.7 applies to the predicate $\mathcal{P}^{=1}(I[A,\ell,u]{=}1)$ which has been considered in [16] for finite-state Markov decision processes.

References

[1] P. Abdulla, N.B. Henda, and R. Mayr. Verifying infinite Markov chains with a finite attractor or the global coarseness property. In *Proceedings of LICS 2005*, pp. 127–136. IEEE, 2005.

[2] 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.

[3] R. Alur, S. Chaudhuri, K. Etessami, and P. Madhusudan. On-the-fly reachability and cycle detection for recursive state machines. In *Proceedings of TACAS 2005*, vol. 3440 of *LNCS*, pp. 61–76. Springer, 2005.

[4] 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.

[5] R. Alur and P. Madhusudan. Visibly pushdown languages. In *Proceedings of STOC 2004*, pp. 202–211. ACM Press, 2004.

[6] 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.

[7] T. Ball and S.K. Rajamani. Bebop: A symbolic model checker for boolean programs. In *SPIN 00: SPIN Workshop*, vol. 1885 of *LNCS*, pp. 113–130. Springer, 2000.

[8] T. Ball and S.K. Rajamani. The SLAM project: debugging system software via static analysis. In *Proceedings of POPL 2002*, pp. 1–3. ACM Press, 2002.

[9] 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.

[10] T. Brázdil, J. Esparza, and A. Kučera. Analysis and prediction of the long-run behavior of probabilistic sequential programs with recursion. Technical report FIMU-RS-2005-08, Faculty of Informatics, Masaryk University, 2005.

[11] T. Brázdil, A. Kučera, and O. Stražovský. On the decidability of temporal properties of probabilistic pushdown automata. In *Proceedings of STACS'2005*, vol. 3404 of *LNCS*, pp. 145–157. Springer, 2005.

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

[13] C. Courcoubetis and M. Yannakakis. Markov decision processes and regular events. In *Proceedings of ICALP'90*, vol. 443 of *LNCS*, pp. 336–349. Springer, 1990.

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

[15] L. de Alfaro. Temporal logics for the specification of performance and reliability. In *Proceedings of STACS'97*, vol. 1200 of *LNCS*, pp. 165–176. Springer, 1997.

[16] L. de Alfaro. How to specify and verify the long-run average behavior of probabilistic systems. In *Proceedings of LICS'98*, pp. 454–465. IEEE, 1998.

[17] J. Esparza, D. Hansel, P. Rossmanith, and S. Schwoon. Efficient algorithms for model checking pushdown systems. In *Proceedings of CAV 2000*, vol. 1855 of *LNCS*, pp. 232–247. Springer, 2000.

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

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

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

[21] K. Etessami and M. Yannakakis. Algorithmic verification of recursive probabilistic systems. In *Proceedings of TACAS 2005*, vol. 3440 of *LNCS*, pp. 253–270. Springer, 2005.

[22] K. Etessami and M. Yannakakis. Checking LTL properties of recursive Markov chains. In *Proceedings of 2nd Int. Conf. on Quantitative Evaluation of Systems (QEST'05)*, 2005.

[23] K. Etessami and M. Yannakakis. Recursive Markov chains, stochastic grammars, and monotone systems of non-linear equations. In *Proceedings of STACS'2005*, vol. 3404 of *LNCS*, pp. 340–352. Springer, 2005.

[24] K. Etessami and M. Yannakakis. Recursive Markov decision processes and recursive stochastic games. In *Proceedings of ICALP 2005*, LNCS. Springer, 2005.

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

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

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

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

[29] A. Remke, B.R. Haverkort, and L. Cloth. Model checking infinite-state Markov chains. In *Proceedings of TACAS 2005*, vol. 3440 of *LNCS*, pp. 237–252. Springer, 2005.

[30] A. Tarski. *A Decision Method for Elementary Algebra and Geometry*. Univ. of California Press, Berkeley, 1951.

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

[32] I. Walukiewicz. Pushdown processes: Games and model-checking. *I&C*, 164(2):234–263, 2001.