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

Tomáš Brázdil*,
Faculty of Informatics, Masaryk University,
Botanická 68a, 60200 Brno,
Czech Republic.
brazdil@fi.muni.cz

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

Antonín Kučera†
Faculty of Informatics, Masaryk University,
Botanická 68a, 60200 Brno,
Czech Republic.
tony@fi.muni.cz

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 [24]. The underlying semantic model for these systems 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* [22]. 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 and implemented mainly for finite-state Markov chains and finite-state Markov decision processes [13, 28, 22, 12, 14]. 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, 23, 25, 2]. Examples of more generic results are [1, 26].

Very recently, probabilistic aspects of recursive sequential programs have also been taken into account. In the non-probabilistic setting, the literature offers two natural models for such programs:

- *pushdown automata (PDA)*, see e.g. [16, 19, 29, 5], where the stack symbols correspond to individual procedures, and the data and control flow 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.

* Supported by the Czech Science Foundation, grant No. 201/03/1161.
† Supported by the Alexander von Humboldt Foundation and by the research centre Institute for Theoretical Computer Science (ITI), project No. 1M0021620808.

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 a successful application of software tools which are based on the designed algorithms [7, 8].

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

In [17], it was shown that the generalized random walk problem for Markov chains generated by probabilistic PDA is decidable, and that the quantitative model-checking problem for deterministic Büchi specifications is also decidable. This study was continued in [10], 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 was studied in greater detail in [21]. In particular, it was shown that the qualitative reachability problem (i.e., the question whether the probability of reaching one 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 [20]. In particular, it was shown that the qualitative variant of this problem is **EXPTIME**-complete. In [18], 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 (not even for finite-state systems). What we are interested in here are *limit properties* of runs and ways how these properties can efficiently be *predicted* after performing (and observing) a bounded initial prefix of a run.

An important source of initial inspiration for this study was the work of Luca de Alfaro presented in [15]. In [15], it is convincingly argued that conventional temporal logics cannot express properties related to long-run average behaviour of probabilistic systems, which include many relevant performance and reliability issues. To get some intuition, consider a system which repeatedly services certain requests (as a concrete example one can take a www server, an answering machine, a telephone switchboard, etc.) The typical problems of performance analysis include 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?", etc. Such properties are not directly expressible in conventional temporal logics. In [15], 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 state 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. For example, one cannot say how much the individual service times deviate from the average service time, i.e., what is the average deviation. If requests with a long service time are for some reason particularly undesirable, one can also be interested what percentage of all services take longer than a given time bound. To be able to formulate such properties, we introduce a family of random variables that capture certain limit properties of runs, and then use these variables to define a family of *run-indicators*. A run-indicator classifies each run as "good" or "bad" according to some criterion, and one can thus formulate questions about the probability of good/bad behaviour. 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 7 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). In this general setting, we provide positive decidability results for the class of Markov chains generated by probabilistic PDA, and a class of reward functions whose values depend both on the current control state and the current stack content of a given PDA configuration. We show that the problem whether $\mathcal{P}(I{=}1) \sim \varrho$, where I is one of the introduced run-indicators, $\mathcal{P}(I{=}1)$ the probability that I is satisfied, $\varrho \in [0,1]$ a rational constant, and $\sim \in \{<, \leq, >, \geq, =\}$, is decidable. This allows to approximate the value of $\mathcal{P}(I{=}1)$ by rational lower and upper

bounds that are arbitrarily close (as we shall see, $\mathcal{P}(I{=}1)$ can be irrational).

Another issue addressed in this paper is *prediction* of the aforementioned limit features. To the best of our knowledge, this problem has not yet been taken into account in previous works, and therefore we explain the underlying intuition in greater detail.

In ergodic Markov chains, the aforementioned limit properties of runs take just one value (with probability one), regardless where a run is initiated. For example, the average service time is the same for "almost all" runs, and hence it does not make much sense to predict the average service time because its value is determined from the very beginning. One can still ask "how fast" a run approaches this limit value, but this is a completely different question which is not addressed in this paper. For 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 behaviour of a run 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, although 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. A practical importance of this result is obvious.

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 **??** we concentrate on probabilistic PDA and show how to compute and predict the properties introduced in Section 3. The underlying intuition and a detailed comparison to previous work is given at appropriate places in Sections 2, 3, and **??** after introducing the necessary notions.

2. Preliminaries

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 ω, 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.

It 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 $pX\alpha$ of Δ, 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 fol-

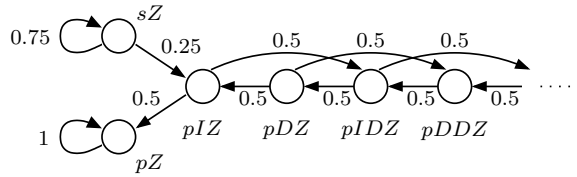

Figure 1. The Markov chain $M_{\bar{\Delta}}$

lowing 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 behaviour 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 $\{w \in Run(sZ) \mid w(i) = pZ \text{ for some } i \in \mathbb{N}_0\}$). In this particular case, we can rely on standard results about one-dimensional random walks and answer that this probability equals $(\sqrt{5}-1)/2$ (this irrational number is commonly known as the "golden cut"). This shows that the quantities of our interest can take irrational values and cannot be computed precisely in general.

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 [17, 21], 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 [27], 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 [11] and conclude that the problem whether $\mathcal{P}(p\alpha \rightarrow^* q\beta) \sim \varrho$ belongs to **PSPACE**.

Observe that once a certain quantity (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 quantities. As long as these expressions contain just multiplication, addition, and inequality over reals, they can again be encoded into $(\mathbb{R}, +, *, \leq)$. Since no quantities are actually *evaluated* during this process, there is no loss of precision and the newly expressed quantities enjoy essentially the same features as the "old" ones. In particular, they can be used as known constants when expressing other quantities, and their values can be effectively approximated. This approach has been used in [18] to express the conditional expected number of transitions needed to reach a configuration $q\beta$ from $p\alpha$ under the condition that $q\beta$ is indeed reached from $p\alpha$. In this case, the "known" quantities are certain probabilities of the form $\mathcal{P}(p\alpha \rightarrow^* q\beta)$.

The previous two paragraphs indicate how to deal with irrational quantities. 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. This means that we cannot directly apply the results of rich theory of ergodic Markov chains, although the problems we are interested in here are typically solved using these methods (see Section 3). This is overcome by abstracting the Markov chain M_Δ into a finite-state Markov chain X_Δ so that certain quantitative properties of M_Δ can be solved by examining the properties of X_Δ. The definition and furher 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 $M = (S, \rightarrow, Prob)$ 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 average 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 *final* states. Let $w \in Run(s_0)$ be a run with infinitely many final states $w(i_1), w(i_2), \ldots$, 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 $j{-}1^{th}$ final state (not included) and the j^{th} final state (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 setup so that the states visited before the actual start of the service are ignored (i.e., have zero reward). Alternatively, one could also consider two families of "on" and "off" states, but the current setup is technically more

convenient and equivalently powerful. 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 $I : Run(s_0) \to \{1, 0\}$ 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:

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

We are primarily interested in those runs w where $I_{inf}(w) = 1$, 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 behaviour 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 [17] 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 *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 efficiently computable just by inspecting the prefix of w of length n.*[1]

Hence, G^n "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 many complicated indicators are well-predictable for a class of pPDA that satisfy a mild and effectively checkable condition.

Now we define other random variables and the associated indicators.

$$A(w) = \begin{cases} \lim_{n\to\infty} \frac{\sum_{j=1}^{n} f(w[j])}{n} & \text{if } I_{inf}(w) = 1 \\ & \text{and the limit exists;} \\ \bot & \text{otherwise.} \end{cases}$$

$$D[\kappa](w) = \begin{cases} \lim_{n\to\infty} \frac{\sum_{j=1}^{n} |f(w[j]) - \kappa|}{n} & \text{if } I_{inf}(w) = 1; \\ & \text{and the limit exists;} \\ \bot & \text{otherwise.} \end{cases}$$

$$R[\lambda, \gamma](w) = \begin{cases} \lim_{n\to\infty} \frac{\sum_{j=1}^{n} B(w[j], \lambda, \gamma)}{n} & \text{if } I_{inf}(w) = 1; \\ & \text{and the limit exists;} \\ \bot & \text{otherwise.} \end{cases}$$

Here $\kappa \in \mathbb{R}$, $\lambda, \gamma \in \mathbb{R}^{\pm\omega}$, and $B(w[j], \lambda, \gamma)$ returns either 1 or 0 depending on whether $\lambda \leq f(w[j]) \leq \gamma$ or not, respectively.

The random 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 centre κ in a given run. Finally, the variable $R[\lambda, \gamma]$ returns the percentage of services whose rewards are within the bounds λ, γ.

In general, $\mathcal{P}(I_{inf}{=}1 \wedge V{=}\bot)$, where V is one of the random variables introduced above, can be positive. However, this cannot happen for Markov chains generated by pPDA, as we shall see in the next section.

Let V be one of the above defined variables, and let $\ell, u \in [0, 1]$. The indicator $I[V, \ell, u]$ is defined as follows:

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

Apart of the aforementioned indicators, we also consider their "Boolean combinations". Thus, we obtain the family $\mathcal{I}$ of indicators, which consists of I_{inf}, all $I[V, \ell, u]$, and their Boolean combinations. To get some intuition why it makes sense to consider Boolean combinations of "basic" indicators, let us formalize the properties mentioned in Section 1 (assume that the reward function corresponds to average 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.
- $I(A, 30, 32) \wedge I(D[31], 0, 5) \wedge I(R[7, \omega], 0, 20\%)$ defines all runs satisfying the previous condition and the condition that the percentage of services longer than 7 is at most 20.

This example is by no means "exhaustive", one can easily formulate other properties (possibly using different reward functions) that are definable in terms of these indicators.

Let $I \in \mathcal{I}$ be an indicator. There are two basic algorithmic problems:

- compute $\mathcal{P}(I{=}1)$;
- check whether $\mathcal{P}(I{=}1) \sim \varrho$ for a given rational constant ϱ and $\sim \in \{<, \leq, >, \geq, =\}$.

1 Due to the Markov property, the last state of the prefi x 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 prefi x. However, this inspection can make the prediction more *effi cient*, as we shall see in Section 4.

The predicate $\mathcal{P}(I{=}1) \sim \varrho$ is either valid or invalid in each state $s \in S$, and hence it can be "plugged" into state-based temporal logics such as LTL, PCTL, or PCTL* in the style of [15] (the state predicate which has been introduced and studied in [15] corresponds to $\mathcal{P}(I[A,\ell,u]{=}1) = 1$). Note that the conditional probability $\mathcal{P}(I{=}1 \mid I_{inf}(w){=}1)$ (which is relevant in situations when $\mathcal{P}(I_{inf}(w){=}1) < 1$) is expressible from the probabilites given above, and hence it does not require a special attention.

In the next section we examine these algorithmic problems for pPDA and a general class of reward functions that take into account both the current control state and the current stack content. We also show that the introduced indicators are well-predictable. All these results work under some mild and effectively checkable conditions, which are formulated and explained in Section **??**. Finally, we show that if *qualitative* state predicates (i.e., state predicates of the form $\mathcal{P}(I{=}1) = 1$) are plugged into LTL or the qualitative fragment of PECTL*, then the model-checking problem for these logics is still decidable. For general predicates, we derive undecidability results.

4. Results for pPDA

Let us fix a pPDA $\Delta = (Q, \Gamma, \delta, Prob)$ and its initial configuration $q_0 Z_0$ 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). We also fix a subset $F \subseteq Q$ of *final* control states, and declare a configuration $p\alpha \in \mathcal{C}(\Delta)$ as final 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 functions $g, h : Q \to \mathbb{R}$ 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. For technical convenience, we assume that $g(p) \cdot h(p) \geq 0$ for all $p \in Q$.*

We say that f is simple *(or* linear*) iff $h(p) = 0$ (or $h(p) = 1$) 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.

Simple reward functions can model gains and costs which do not depend on the history of activation records. A simple example is execution time—one can reasonably assume that the expected 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 might wish not to "count" certain 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 we are interested in, and hence the obtained results can become irrelevant. However, using h one can "switch off" the intermediate states so that they do not contribute to the accumulated reward.

As already menioned in Section **??**, Markov chains generated by pPDA are not necessarily ergodic. Nevertheless, questions about long-run avarge behaviour are inherently related to concepts of ergodic chains (in particular, stationary distributions would be very useful in here). Fortunatelly, 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. This construction is essentially due to [17]. Here we introduce a 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 Markov chain M_Δ is introduced in the next subsection.

The Markov chain $\mathbf{X_\Delta}$. Let $w = p_1\alpha_1, p_2\alpha_2 \cdots$ be a run of $Run(q_0 Z_0)$ (notice that $p_1\alpha_1 = q_0 Z_0$). For each $i \in \mathbb{N}$ we define the i^{th} *minimum* of w, denoted $\min_i(w)$, which is either *increasing* or *non-increasing*. The definition is inductive.

- $\min_1(w) = p_1\alpha_1$ (i.e., $\min_1(w)$ is the starting configuration $q_0 Z_0$ of w). We stipulate that $\min_1(w)$ is non-increasing.
- Let $\min_i(w) = p_\ell\alpha_\ell$. Then $\min_{i+1}(w) = p_k\alpha_k$ where k is the least number such that $k > \ell$ and $|\alpha_{k'}| \geq |\alpha_k|$ for each $k' \geq k$. Observe that $|\alpha_k| - |\alpha_\ell|$ equals either 1 or 0. In the first case, $\min_{i+1}(w)$ is increasing. Otherwise, $\min_{i+1}(w)$ 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. In other words, the future behaviour of two configurations of the form $pY\alpha$ and $pY\beta$ is the same, assuming that these configurations are

minimal. This intuition is formally captured in our next definitions.

For all $p, q \in Q$ and $Y \in \Gamma$, we use $[pXq]$ to abbreviate $\mathcal{P}(pX \rightarrow^* q\varepsilon)$, and $[pX\uparrow]$ to abbreviate $1 - \sum_{r \in Q}[pXr]$. Hence, $[pX\uparrow]$ is the probability that the stack never becomes empty along a run initiated in pX.

For each $i \in \mathbb{N}$ we define a random variable X_i over $Run(q_0Z_0)$ as follows: $X_i(w) = (qY, m)$, 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 [17], 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 $\mathcal{P}(\bigwedge_{i=1}^{n-1} X_i{=}(q_iY_i, m_i)){>}0$ we have that the probability

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

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

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

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 it is also independent of the value of n. Hence, we can define a finite-state Markov chain X_Δ whose states are pairs of the form (qY, m) and the probability of $(qY, m) \rightarrow (q'Y', m')$ 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, $(qY, m) \rightarrow (q'Y', m')$ 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 $(\mathbb{R}, +, *, \leq)$, we can conclude that the transition probabilities of X_Δ are also effectively expressible in $(\mathbb{R}, +, *, \leq)$ (cf. the previous paragraph).

As an example, consider again the pPDA $\bar{\Delta}$ defined in Section 2, where sZ plays the role of initial configuration. 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 $w \in Run(q_0Z_0)$ we associate its *footprint* $X_1(w), X_2(w), \cdots$. Note that there can be 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 (here we view X_Δ as a finite graph where the edges correspond to transitions of X_Δ). To each $C \in BSCC_\Delta$ we associate the set $Run(q_0Z_0, C)$ consisting of all $w \in Run(q_0Z_0)$ 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 $w \in Run(q_0Z_0)$ returns either $w(j)$ where $j \in \mathbb{N}$ is the least number such that $X_j(w) \in C$ for some $C \in BSCC_\Delta$, or $\perp$ if there is no such j. In other words, if w is a run whose footprint hits a BSCC of X_Δ, then $Entry(w)$ is the configuration which "enters" this BSCC.

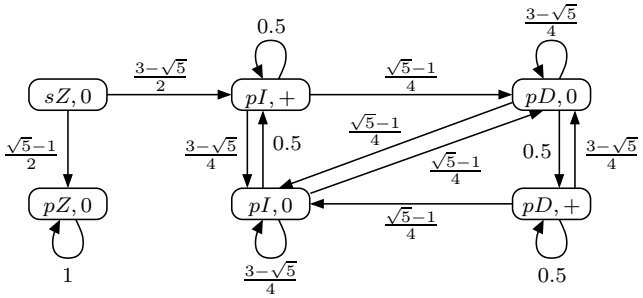

Figure 2. The Markov chain $X_{\bar{\Delta}}$

Note that since X_Δ has finitely many states, $\mathcal{P}(Run(q_0Z_0, C_i))$ 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

$$\sum_{C \in BSCC_\Delta} \mathcal{P}(Run(q_0Z_0, C)) = 1 \qquad (1)$$

Consequently, $\mathcal{P}(Entry{=}\perp) = 0$.

Solving the Problems of Section 3 for pPDA. Now we formulate a crucial result which says that the (in)validity of all indicators in our family $\mathcal{I}$ for a given $w \in Run(q_0Z_0)$ is essentially determined only by the BSCC of X_Δ hit by w, and by the stack content in the configurations whis enters this component. To formulate this precisely, we need to introduce an 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 f a well-defined reward function. For every $C \in BSCC_\Delta$ there effectively exists a regular language $L_C \subseteq \Gamma^*$ such that $\mathcal{P}(I = Hit_{L_C} \mid Run(q_0Z_0, C)) = 1$. Moreover, if f is a simple reward function, then $L_C = \Gamma^*$ for each C.*

The proof of Theorem 4.2 is the technical core of our results.

Theorem 4.3. *Let $L \subseteq \Gamma^*$ be a regular language and $C \in BSCC_\Delta$. Then $\mathcal{P}(Hit_L{=}1 \mid Run(q_0 Z_0, C))$ is effectively expressible in $(\mathbb{R}, +, *, \leq)$. Moreover, if $L = \Gamma^*$, then the size of the constructed formula is only polynomial in the size of Δ, and the alternation depth of quantifiers is fixed.*

As a direct corollary of Theorem 4.2 and Theorem 4.3 we obtain the following:

Corollary 4.4. *Let $I \in \mathcal{I}$ be an indicator, f a well-defined reward function, ϱ a rational constant, and $\sim \in \{<, \leq, >, \geq, =\}$. The problem whether $\mathcal{P}(I{=}1) \sim \varrho$ is decidable. Moreover, if f is a simple reward function, then this problem belongs to **EXPTIME**.*

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*. IEEE, 2005. To appear.

[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. Br´azdil, A. Kučera, and O. Stražovsk´y. On the decidability of temporal properties of probabilistic pushdown automata. In *Proceedings of STACS'2005*, vol. 3404 of *LNCS*, pp. 145–157. Springer, 2005.

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

[12] 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.

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

[14] 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.

[15] 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.

[16] 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.

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

[18] J. Esparza, A. Kučera, and R. Mayr. Quantitative analysis of probabilistic pushdown automata: Expectations and variances. In *Proceedings of LICS 2005*. IEEE, 2005. To appear.

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

[20] 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.

[21] 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.

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

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

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

[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] 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.

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

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

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