LIPIcs Leibniz International Proceedings in Informatics

On the Memory Consumption of Probabilistic Pushdown Automata *

Tomáš Brázdil[1], **Javier Esparza**[2], **Stefan Kiefer**[2]

[1] Faculty of Informatics, Masaryk University, Brno, Czech Republic
xbrazdil@fi.muni.cz

[2] Institut für Informatik, Technische Universität München, Germany
{esparza,kiefer}@in.tum.de

ABSTRACT. We investigate the problem of evaluating memory consumption for systems modelled by probabilistic pushdown automata (pPDA). The space needed by a run of a pPDA is the maximal height reached by the stack during the run. The problem is motivated by the investigation of depth-first computations that play an important role for space-efficient schedulings of multithreaded programs.

We study the computation of both the distribution of the memory consumption and its expectation. For the distribution, we show that a naive method incurs an exponential blow-up, and that it can be avoided using linear equation systems. We also suggest a possibly even faster approximation method. Given $\varepsilon > 0$, these methods allow to compute bounds on the memory consumption that are exceeded with a probability of at most ε.

For the expected memory consumption, we show that whether it is infinite can be decided in polynomial time for stateless pPDA (pBPA) and in polynomial space for pPDA. We also provide an iterative method for approximating the expectation. We show how to compute error bounds of our approximation method and analyze its convergence speed. We prove that our method converges linearly, i.e., the number of accurate bits of the approximation is a linear function of the number of iterations.

1 Introduction

The verification of probabilistic programs with possibly recursive procedures has been intensely studied in the last years. The Markov chains or Markov Decision Processes underlying these systems may have infinitely many states. Despite this fact, which prevents the direct application of the rich theory of finite Markov Chains, many positive results have been obtained. Model-checking algorithms have been proposed for both linear and branching temporal logics [8, 11, 18], the long-run behavior of the systems has been analyzed [6, 9], and algorithms deciding properties of games have been described (see e.g. [10]).

In all these papers programs are modelled as probabilistic pushdown automata (pPDA) or as recursive Markov chains; the two models are very close, and nearly all results obtained for one of them can be easily translated to the other [7]. In this paper we consider pPDA. Loosely speaking, a pPDA is a pushdown automaton whose transitions carry probabilities. The *configurations* of a pPDA are pairs containing the current control state and the current stack content. A *run* is a sequence of configurations, each one obtained from its predecessor

*The first author was supported by the research center Institute for Theoretical Computer Science (ITI), project No. 1M0545.

by applying a transition, which may modify the control state and modify the top of the stack. If a run reaches a configuration with empty stack, it stays in this configuration forever. We say "it terminates".

The memory consumption of a pPDA is modelled by the random variable M that assigns to a run the *maximal* stack height of the configurations visited along it (which may be infinite). We study the distribution and the expected value of M. The execution time and memory consumption of pPDA were studied in [9], but the results about the latter were much weaker. More precisely, all it was shown in [9] was that $\mathcal{P}(M = \infty)$ can be compared with 0 or 1 in polynomial space and with other rationals in exponential time.

A probabilistic recursive program whose variables have finite range can be modelled by a pPDA, and in this case M models the amount of memory needed for the recursion stack. But M is also an important parameter for the problem of scheduling multithreaded computations (see [15, 3] among other papers). When a multithreaded program is executed by one processor, a scheduler decides which thread to execute next, and the current states of all other active threads are stored. When threads are lightweight, this makes the memory requirements of the program heavily depend on the thread scheduler [15]. Under the usual assumption that a thread can terminate only after all threads spawned by it terminate, the space-optimal scheduler is the one that, when A spawns B, interrupts the execution of A and continues with B; this is called the *depth-first scheduler* in [15, 3]. The depth-first scheduler can be modelled by a pushdown automaton. To give an example, consider a multithreaded system with two types of threads, X and Y. Imagine that through statistical sampling we know that a thread of type X spawns either a thread of type Y or no new threads, both with probability $1/2$; a thread of type Y spawns another thread of type Y with probability $1/3$, or no new thread with probability $2/3$. The depth-first execution of this multithreaded program is modelled by a pPDA with one control state, stack symbols X, Y, and rules $X \stackrel{1/2}{\to} YX$, $X \stackrel{1/2}{\to} \varepsilon$, $Y \stackrel{1/3}{\to} YY$, $Y \stackrel{2/3}{\to} \varepsilon$. Notice that the rule $X \stackrel{1/2}{\to} YX$ indeed models the depth-first idea: the new thread Y is executed before the thread X.

In this simple model, pPDA for multithreaded systems have one single control state. Stack symbols represent currently active threads and pushdown rules model whether a thread dies or spawns a new thread. On the other hand, pPDA with more than one control state can model global variables with finite range (the possible values of the global store are encoded into the control states of the pPDA) [4]. For these reasons we study arbitrary pPDA in this paper, but also specialize our results (and in particular the complexity of algorithms) to so-called pBPA, which are pPDA with a single control state. As we shall see, while some algorithms are polynomial for pBPA, this is unlikely to be the case for pPDA.

Our contribution. We specifically address the problem of computing $\mathcal{P}(M \geq n)$, or at least an upper bound, for a given n. This allows us to determine the size that the stack (or the store for threads awaiting execution) must have in order to guarantee that the probability of a memory overflow does not exceed a given bound. In Section 3 we obtain a system of recurrence equations for $\mathcal{P}(M \geq n)$, and show that for a pPDA with set Q of control states and set Γ of stack symbols, $\mathcal{P}(M \geq n)$ can be computed in time $\mathcal{O}\left(n \cdot (|Q|^2 \cdot |\Gamma|)^3\right)$ (time $\mathcal{O}\left(n \cdot |\Gamma|^3\right)$ for pBPAs) in the Blum-Shub-Smale model, the computation model in which an arithmetic operation takes one time unit, independently of the size of the operands. How-

ever, this result does not provide any information on the asymptotic behavior of $\mathcal{P}(M \geq n)$ when n grows, and moreover the algorithm is computationally inefficient for large values of n. We address these problems for pPDA in which the expected value of M is finite. We show in Section 3.2 that in this case $\mathcal{P}(M \geq n) \in \Theta(\rho^n)$, where $\rho < 1$ is the spectral radius of a certain matrix. This determines the exact asymptotic behavior up to a constant, and also leads to an algorithm for computing a bound on $\mathcal{P}(M \geq n)$ with logarithmic runtime in n.

Then we turn to computing the expectation of M. In Section 3.3 we provide an algorithm that approximates the expectation, give an error bound and analyze its convergence speed. Finally, in Section 4 we study the problem of deciding whether the expected value of M is finite. We show that the problem is polynomial for pBPAs. For arbitrary pPDA we show that the problem is in PSPACE and at least as hard as the SQRT-SUM and PosSLP problems. Notice that already the problem of deciding if the termination probability of a pPDA exceeds a given bound has this same complexity.

The full version of this paper [5] includes the proofs and more discussion.

Related work. Much work has been done also on the analysis of other well-structured infinite-state Markov chains, such as quasi-birth-death processes and Jackson queueing networks [16] and probabilistic lossy channel systems [17]. However, none of these classes contain pPDA or even pBPA. There is also work on general infinite-state (continuous-time) Markov chains which analyzes the behavior of the chain up to a finite depth [12]. This method is very general, but it is inefficient for pPDA, because it has not been designed to exploit the pushdown structure. Our analysis techniques are strongly based on linear algebra and matrix theory, in particular Perron-Frobenius theory [2]. The closest work to ours is [11] which also uses Perron-Frobenius theory for assessing the termination probability of recursive Markov chains.

2 Preliminaries

In the rest of this paper, $\mathbb{N}$ and $\mathbb{R}$ denote the set of positive integers and real numbers, respectively. The set of all finite words over a given alphabet Σ is denoted by Σ^*, and the set of all infinite words over Σ is denoted by Σ^ω. We write ε for the empty word. Given two sets $K \subseteq \Sigma^*$ and $L \subseteq \Sigma^* \cup \Sigma^\omega$, we use $K \cdot L$ (or just KL) to denote the concatenation of K and L, i.e., $KL = \{ww' \mid w \in K, w' \in L\}$. The length of a given $w \in \Sigma^* \cup \Sigma^\omega$ is denoted by $|w|$, where the length of an infinite word is ∞. Given a word (finite or infinite) over Σ, the individual letters of w are denoted by $w(0), w(1), \ldots$

Vectors and Matrices. Given a set S, we regard the elements of $\mathbb{R}^S$ as *vectors*. We use bold letters, like $\mathbf{u}$, for vectors. The vector whose components are all 0 (resp. all 1) is denoted by $\mathbf{0}$ (resp. $\mathbf{1}$). We write $\mathbf{u} = \mathbf{v}$ (resp. $\mathbf{u} \leq \mathbf{v}$) if $\mathbf{u}(s) = \mathbf{v}(s)$ (resp. $\mathbf{u}(s) \leq \mathbf{v}(s)$) holds for all $s \in S$. If $S' \subseteq S$, we write $\mathbf{u}_{|S'}$ for the vector $\mathbf{v} \in \mathbb{R}^{S'}$ with $\mathbf{v}(s) = \mathbf{u}(s)$ for all $s \in S'$.

Given two vector spaces $\mathbb{R}^S, \mathbb{R}^T$ we identify a linear function $A : \mathbb{R}^S \to \mathbb{R}^T$ with its corresponding matrix $A \in \mathbb{R}^{T \times S}$. We use capital letters for matrices and I for the identity matrix. We call a matrix nonnegative if all its entries are nonnegative. For nonnegative square matrices $A \in \mathbb{R}^{S \times S}$ we define the matrix sum $A^* = \sum_{i=0}^{\infty} A^i = I + A + AA + \cdots$. It is a well-known fact (see e.g. [13]) that A^* converges (or "exists") iff $\rho(A) < 1$, where

$\rho(A)$ denotes the spectral radius of A, i.e., the largest absolute value of the eigenvalues of A. Perron-Frobenius theory for nonnegative matrices (see e.g. [2]) states that $\rho(A)$ is an eigenvalue of A. If A^* exists, then $A^* = (I - A)^{-1}$.

Markov Chains. Our models of interest induce (infinite-state) Markov chains.

DEFINITION 1. *A* 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 $Prob(s \to t) > 0$ so that for every $s \in S$ we have $\sum_{s \to t} Prob(s \to t) = 1$ (as usual, we write $s \xrightarrow{x} t$ instead of $Prob(s \to t) = x$).*

A *path* in M is a finite or infinite word $w \in S^+ \cup S^\omega$ such that $w(i-1) \to w(i)$ for every $1 \leq i < |w|$. A *run* in M is an infinite path in M. We denote by $Run[M]$ the set of all runs in M. The set of all runs that start with a given finite path w is denoted by $Run[M](w)$. When M is understood, we write Run (or $Run(w)$) instead of $Run[M]$ (or $Run[M](w)$, resp.).

To every $s \in S$ we associate the probability space $(Run(s), \mathcal{F}, \mathcal{P})$ where $\mathcal{F}$ is the σ-field generated by all *basic cylinders* $Run(w)$ where w is a finite path starting with s, and $\mathcal{P} : \mathcal{F} \to [0,1]$ is the unique probability measure such that $\mathcal{P}(Run(w)) = \Pi_{i=1}^{|w|-1} x_i$ where $w(i-1) \xrightarrow{x_i} w(i)$ for every $1 \leq i < |w|$. If $|w| = 1$, we put $\mathcal{P}(Run(w)) = 1$. Only certain subsets of $Run(s)$ are $\mathcal{P}$-measurable, but in this paper we only deal with "safe" subsets that are guaranteed to be in $\mathcal{F}$. Given $s \in S$ and $A \subseteq S$, we say *A is reachable from s* if $\mathcal{P}(\{w \in Run(s) \mid \exists i \geq 0 : w(i) \in A\}) > 0$.

Probabilistic Pushdown Automata (pPDA).

DEFINITION 2. *A* probabilistic pushdown automaton (pPDA) *is a tuple $\Delta = (Q, \Gamma, \delta, Prob)$ where Q is a finite set of* control states*, Γ is a finite* stack alphabet*, $\delta \subseteq Q \times \Gamma \times Q \times \Gamma^{\leq 2}$ (where $\Gamma^{\leq 2} = \{\alpha \in \Gamma^*, |\alpha| \leq 2\}$) is a* transition relation*, and $Prob$ is a function which to each transition $pX \to q\alpha$ assigns a rational probability $Prob(pX \to q\alpha) > 0$ so that for all $p \in Q$ and $X \in \Gamma$ we have that $\sum_{pX \to q\alpha} Prob(pX \to q\alpha) = 1$ (as usual, we write $pX \xrightarrow{x} q\alpha$ instead of $Prob(pX \to q\alpha) = x$).*

Elements of $Q \times \Gamma^*$ are called *configurations* of Δ. A pPDA with just one control state is called pBPA (pBPAs correspond to 1-exit recursive Markov chains defined in [11]). In what follows, configurations of pBPAs are usually written without the control state (i.e., we write only α instead of $p\alpha$).

EXAMPLE 3 *As a running example we choose the pBPA $\Delta = (\{p\}, \{X, Y, Z, W\}, \delta, Prob)$ with δ and $Prob$ given as follows.*

$$\begin{array}{llll} X \xrightarrow{1/4} \varepsilon & X \xrightarrow{1/4} Y & Y \xrightarrow{2/3} \varepsilon & Z \xrightarrow{1} Z \\ X \xrightarrow{1/4} XX & X \xrightarrow{1/4} Z & Y \xrightarrow{1/3} YY & W \xrightarrow{1} YW \end{array}$$

We can interpret this example as a model of a multithreaded system with four kinds of threads. Notice that threads of type Z and W do not terminate (our results also deal with this possibility). We are interested in the minimal number of threads n such that the probability that the execution of X requires to store more than n threads is at most 10^{-5}.

We define the size $|\Delta|$ of a pPDA Δ as follows: $|\Delta| = |Q| + |\Gamma| + |\delta| + |Prob|$ where $|Prob|$ equals the sum of sizes of binary representations of values of $Prob$. To Δ we associate the Markov chain M_Δ with $Q \times \Gamma^*$ as set of states and transitions defined as follows:

- $p\varepsilon \xrightarrow{1} p\varepsilon$ for each $p \in Q$;
- $pX\beta \xrightarrow{x} q\alpha\beta$ is a transition of M_Δ iff $pX \xrightarrow{x} q\alpha$ is a transition of Δ.

Given $p, q \in Q$ and $X \in \Gamma$, we often write pXq to denote (p, X, q). Given pXq we define

$$Run(pXq) = \{w \in Run(pX) \mid \exists i \geq 0 : w(i) = q\varepsilon\} \quad \text{and} \quad [pXq] = \mathcal{P}(Run(pXq)) .$$

Maximal Stack Height. Given $p\alpha \in Q \times \Gamma^*$, we denote by $height(p\alpha) = |\alpha|$ the stack height of $p\alpha$. Given $pX \in Q \times \Gamma$, the maximal stack height of a run is defined by setting

$$M_{pX}(w) = \sup\{height(w(i)) \mid i \geq 0\} \quad \text{for all runs } w \in Run(pX).$$

It is easy to show that for all $n \in \mathbb{N} \cup \{\infty\}$ the set $M_{pX}^{-1}(n) = \{w \in Run(pX) \mid M_{pX}(w) = n\}$ is measurable. Hence the expectation EM_{pX} of M_{pX} exists and we have

$$EM_{pX} = \sum_{n \in \mathbb{N} \cup \{\infty\}} n \cdot \mathcal{P}(M_{pX}^{-1}(n)) .$$

For what follows, we fix a pPDA $\Delta = (Q, \Gamma, \delta, Prob)$ with initial configuration $p_0X_0 \in Q \times \Gamma$. We are interested in the random variable $M_{p_0X_0}$ modelling the memory consumption of Δ. We wish to compute or approximate the distribution of $M_{p_0X_0}$ and its expectation.

3 Computing the Memory Consumption

The problem of computing the distribution of the maximal stack height is the problem of computing the probability of reaching a given height. So, for every $n \geq 1$ we define a vector $\mathbf{P}[n] \in \mathbb{R}^{Q \times \Gamma}$ with

$$\mathbf{P}[n](pX) = \mathcal{P}(\{w \in Run(pX) \mid M_{pX}(w) \geq n\}) \quad \text{for every } pX \in Q \times \Gamma ,$$

i.e., $\mathbf{P}[n](pX)$ is the probability that the maximal stack height is $\geq n$ in a run of $Run(pX)$.

There is a "naive" method to compute $\mathbf{P}[n](p_0X_0)$. (Recall that M_Δ is the Markov chain associated with Δ.) First, compute the Markov chain M_Δ^{n+1} obtained from M_Δ by restricting it to the states with a height of at most $n+1$. Note that M_Δ^{n+1} has finitely many states. Then compute $\mathbf{P}[n](p_0X_0)$ by computing the probability of reaching a state of height $n+1$ starting from p_0X_0. This can be done as usual by solving a linear equation system. The problem with this approach is that the number of states in M_Δ^{n+1} is $\Theta(|Q| \cdot |\Gamma|^n)$, i.e., exponential in n, and the linear equation system has equally many equations.

A better algorithm is obtained by observing that the Markov chain induced by a pPDA has a certain regular structure. We exploit this to get rid of the state explosion in the "naive" method. (This has also been observed in the analysis of other structured infinite-state systems, see e.g. [16].) In the following we describe the improved method, which is based on linear recurrences. We are mainly interested in the probabilities $\mathbf{P}[n]$ to reach height n, but as

an auxiliary quantity we use the probability of not exceeding height n in terminating runs. Formally, for every $n \geq 0$ we define a vector $\mathbf{T}[n] \in \mathbb{R}^{Q \times \Gamma \times Q}$ such that

$$\mathbf{T}[n](pXq) = \mathcal{P}(\{w \in Run(pXq) \mid M_{pX}(w) \leq n\}) \quad \text{for every } pXq \in Q \times \Gamma \times Q\,,$$

i.e., $\mathbf{T}[n](pXq)$ is the probability of all runs of $Run(pX)$ that terminate at q and do not exceed the height n. To every $pXq \in Q \times \Gamma \times Q$ we associate a variable $\mathbf{T}\langle n\rangle(pXq)$. Consider the following equation system: If $\mathbf{T}[n](pXq) = 0$, then we put $\mathbf{T}\langle n\rangle(pXq) = 0$. Otherwise, we put

$$\mathbf{T}\langle n\rangle(pXq) = \sum_{pX \xrightarrow{y} q\varepsilon} y + \sum_{pX \xrightarrow{y} rY} y\mathbf{T}\langle n\rangle(rYq) + \sum_{pX \xrightarrow{y} rYZ} \sum_{s \in Q} y\mathbf{T}[n-1](rYs)\mathbf{T}\langle n\rangle(sZq)\,.$$

PROPOSITION 4. *For $n \geq 0$, the vector $\mathbf{T}[n]$ is the* unique *solution of that equation system.*

The values $\mathbf{T}[n]$ can be used to set up an equation system for $\mathbf{P}[n]$. To every $pX \in Q \times \Gamma$ we associate a variable $\mathbf{P}\langle n\rangle(pX)$. Consider the following equation system: We put $\mathbf{P}\langle 1\rangle(pX) = 1$. If $\mathbf{P}[n](pX) = 0$, then we put $\mathbf{P}\langle n\rangle(pX) = 0$. Otherwise, we put

$$\mathbf{P}\langle n\rangle(pX) = \sum_{pX \xrightarrow{y} qY} y\mathbf{P}\langle n\rangle(qY) + \sum_{pX \xrightarrow{y} qYZ} y\mathbf{P}[n-1](qY) + \sum_{pX \xrightarrow{y} qYZ} \sum_{r \in Q} y\mathbf{T}[n-2](qYr)\mathbf{P}\langle n\rangle(rZ)\,.$$

PROPOSITION 5. *For $n \geq 1$, the vector $\mathbf{P}[n]$ is the* unique *solution of that equation system.*

EXAMPLE 6 *In our example we have for $n \geq 1$*

$$\begin{aligned}
\mathbf{T}[n](X) &= 1/4 + 1/4\ \mathbf{T}[n](Y) + 1/4\ \mathbf{T}[n](Z) + 1/4\ \mathbf{T}[n-1](X)\mathbf{T}[n](X)\\
\mathbf{T}[n](Y) &= 2/3 + 1/3\ \mathbf{T}[n-1](Y)\mathbf{T}[n](Y)\\
\mathbf{T}[n](Z) &= 0\\
\mathbf{T}[n](W) &= 0
\end{aligned}$$

and for $n \geq 2$

$$\begin{aligned}
\mathbf{P}[n](X) &= 1/4\ \mathbf{P}[n](Y) + 1/4\ \mathbf{P}[n](Z) + 1/4\ \mathbf{P}[n-1](X) + 1/4\ \mathbf{T}[n-2](X)\mathbf{P}[n](X)\\
\mathbf{P}[n](Y) &= 1/3\ \mathbf{P}[n-1](Y) + 1/3\ \mathbf{T}[n-2](Y)\mathbf{P}[n](Y)\\
\mathbf{P}[n](Z) &= 0\\
\mathbf{P}[n](W) &= \mathbf{P}[n-1](Y) + \mathbf{T}[n-2](Y)\mathbf{P}[n](W)\,.
\end{aligned}$$

Solving those systems successively for increasing n shows that $n = 17$ is the smallest number n such that $\mathbf{P}[n](X) \leq 10^{-5}$. In the interpretation as a multithreaded system this means that the probability that 17 or more threads need to be stored is at most 10^{-5}.

Using the above equation systems, we can compute $\mathbf{T}[n]$ and $\mathbf{P}[n]$ iteratively for increasing n by plugging in the values obtained in earlier iterations. The cost of each iteration is dominated by solving the equation system for $\mathbf{T}[n]$, which can be done, using Gaussian elimination, in time $\mathcal{O}\left((|Q|^2 \cdot |\Gamma|)^3\right)$ in the Blum-Shub-Smale model. So the total time to compute $\mathbf{P}[n]$ is linear in n.

PROPOSITION 7. *The value* $\mathbf{P}[n]$ *can be computed by setting up and solving the equation systems of Propositions 4 and 5 in time* $\mathcal{O}\left(n \cdot (|Q|^2 \cdot |\Gamma|)^3\right)$ *in the Blum-Shub-Smale model.*

The values $\mathbf{P}[n]$ that can be computed by Proposition 7 also allow to approximate the expectation $EM_{p_0X_0}$: Since $EY = \sum_{n=1}^{\infty} \mathcal{P}(Y \geq n)$ holds for any random variable Y with values in $\mathbb{N}$, we have $EM_{p_0X_0} = \sum_{n=1}^{\infty} \mathbf{P}[n](p_0X_0)$, so one can approximate $EM_{p_0X_0}$ by computing $\sum_{n=1}^{k} \mathbf{P}[n](p_0X_0)$ for some finite k.

Proposition 7 is simple and effective, but not fully satisfying for several reasons. First, it does not indicate how fast $\mathbf{P}[n](p_0X_0)$ decreases (if at all) for increasing n. Second, although computing $\mathbf{P}[n]$ using Proposition 7 is more efficient than using the "naive" method, it may still be too costly for large n, especially if Q or Γ are large. Instead, one may prefer an upper bound on $\mathbf{P}[n]$ if it is fast to compute. Finally, we wish for an approximation method for $EM_{p_0X_0}$ that comes with an error bound.

In the following we achieve these goals for pPDAs in which the expected memory consumption is finite. So we assume the following on the pPDA Δ for the rest of the section.
ASSUMPTION: The expectation $EM_{p_0X_0}$ is finite.
Notice that from the practical point of view this is a mild assumption: systems with infinite expected memory consumption also have infinite expected running time, and are unlikely to be considered suitable in reasonable scenarios. In Section 4 we show that whether $EM_{p_0X_0}$ is finite can be decided in polynomial time for pBPA, but also that this problem is unlikely to be decidable in polynomial time for general pPDA.

3.1 The Matrix A

This subsection leads to a matrix A which is crucial for our analysis. It is useful to get rid of certain irregularities in the equation systems of Propositions 4 and 5. The following lemma shows that the variables in the equation systems do not change from 0 to positive (or from positive to 0) if n is sufficiently large. (Recall that, by definition, $\mathbf{T}[n] \leq \mathbf{T}[n+1]$ and $\mathbf{P}[n] \geq \mathbf{P}[n+1]$ for all $n \geq 1$.)

LEMMA 8.
1. $\mathbf{T}[|Q|^2|\Gamma|+1](pXq) > 0 \iff$ *for all* $n \geq |Q|^2|\Gamma|+1 : \mathbf{T}[n](pXq) > 0 \iff [pXq] > 0$;
2. $\mathbf{P}[|Q||\Gamma|+1](pX) > 0 \iff$ *for all* $n \geq 1 : \mathbf{P}[n](pX) > 0$.

Another irregularity can be removed by restricting $\mathbf{T}[n]$ and $\mathbf{P}[n]$ to their "interesting" components; in particular, we filter out entries of $\mathbf{P}[n]$ that cannot create large stacks. Let $\mathcal{T} \subseteq Q \times \Gamma \times Q$ denote the set of all pXq such that $pX\Gamma^*$ is reachable from p_0X_0, and $[pXq] > 0$. Let $\mathcal{H} \subseteq Q \times \Gamma$ denote the set of all pX such that $pX\Gamma^*$ is reachable from p_0X_0, and $\mathbf{P}[n](pX) > 0$ for all $n \geq 1$.

LEMMA 9. *The sets* $\mathcal{T}$ *and* $\mathcal{H}$ *are computable in polynomial time.*

EXAMPLE 10 *For our running example, we fix X as the initial configuration. Then* $W\Gamma^*$ *is not reachable and* $\mathbf{P}[n](Z) = 0$ *for* $n \geq 2$, *hence* $\mathcal{H} = \{X, Y\}$. *Furthermore,* $\mathcal{T} = \{X, Y\}$.
We define $\mathbf{t}[n] \in \mathbb{R}^{\mathcal{T}}$ by $\mathbf{t}[n] := \mathbf{T}[n]_{|\mathcal{T}}$, i.e., $\mathbf{t}[n] \in \mathbb{R}^{\mathcal{T}}$ is the restriction of $\mathbf{T}[n]$ to $\mathcal{T}$. Similarly, we define $\mathbf{p}[n] := \mathbf{P}[n]_{|\mathcal{H}}$. Now we bring the equation systems for $\mathbf{t}[n]$ and $\mathbf{p}[n]$ from Propositions 4 and 5 in a compact matrix form.

For $\mathbf{t}[n]$, we define a vector $\mathbf{c} \in \mathbb{R}^{\mathcal{T}}$, a linear function $\tilde{L}$ on $\mathbb{R}^{\mathcal{T}}$, and a bilinear function $\tilde{Q} : \mathbb{R}^{\mathcal{T}} \times \mathbb{R}^{\mathcal{T}} \to \mathbb{R}^{\mathcal{T}}$ as follows:

$$(\mathbf{c})(pXq) = \sum_{pX \xrightarrow{y} q\varepsilon} y \qquad (\tilde{L}\mathbf{v})(pXq) = \sum_{pX \xrightarrow{y} rY,\, rYq \in \mathcal{T}} y\mathbf{v}(rYq)$$
$$(\tilde{Q}(\mathbf{u},\mathbf{v}))(pXq) = \sum_{pX \xrightarrow{y} rYZ} \; \sum_{s \in Q,\, rYs \in \mathcal{T},\, sZq \in \mathcal{T}} y\mathbf{u}(rYs)\mathbf{v}(sZq)$$

By $\tilde{Q}(\mathbf{u}, \cdot)$ we denote a linear function satisfying $\tilde{Q}(\mathbf{u}, \cdot)(\mathbf{v}) = \tilde{Q}(\mathbf{u}, \mathbf{v})$.

For $\mathbf{p}[n]$, we define linear functions L and L' on $\mathbb{R}^{\mathcal{H}}$, and a bilinear function $Q : \mathbb{R}^{\mathcal{T}} \times \mathbb{R}^{\mathcal{H}} \to \mathbb{R}^{\mathcal{H}}$ as follows:

$$(L\mathbf{v})(pX) = \sum_{pX \xrightarrow{y} qY,\, qY \in \mathcal{H}} y\mathbf{v}(qY) \qquad (L'\mathbf{v})(pX) = \sum_{pX \xrightarrow{y} qYZ,\, qY \in \mathcal{H}} y\mathbf{v}(qY)$$
$$(Q(\mathbf{u},\mathbf{v}))(pX) = \sum_{pX \xrightarrow{y} qYZ} \; \sum_{r \in Q,\, qYr \in \mathcal{T},\, rZ \in \mathcal{H}} y\mathbf{u}(qYr)\mathbf{v}(rZ)$$

By $Q(\mathbf{u}, \cdot)$ we denote a linear function satisfying $Q(\mathbf{u}, \cdot)(\mathbf{v}) = Q(\mathbf{u}, \mathbf{v})$.
Using Propositions 4 and 5 we obtain for $n \geq |Q|^2|\Gamma| + 3$ (recall Lemma 8):

PROPOSITION 11. *The following equations hold for all $n \geq |Q|^2|\Gamma| + 3$:*

$$\mathbf{t}[n] = \mathbf{c} + \tilde{L}\mathbf{t}[n] + \tilde{Q}(\mathbf{t}[n-1], \mathbf{t}[n]) \quad \text{and} \quad \mathbf{p}[n] = L\mathbf{p}[n] + L'\mathbf{p}[n-1] + Q(\mathbf{t}[n-2], \mathbf{p}[n])$$

EXAMPLE 12 *In our example we have for $n \geq 1$*

$$\mathbf{t}[n] = \overbrace{\begin{pmatrix} 1/4\,\mathbf{t}[n-1](X) & 1/4 \\ 0 & 1/3\,\mathbf{t}[n-1](Y) \end{pmatrix}}^{\tilde{L}+\tilde{Q}(\mathbf{t}[n-1],\cdot)} \mathbf{t}[n] + \overbrace{\begin{pmatrix} 1/4 \\ 2/3 \end{pmatrix}}^{\mathbf{c}}$$

and for $n \geq 2$

$$\mathbf{p}[n] = \overbrace{\begin{pmatrix} 1/4\,\mathbf{t}[n-2](X) & 1/4 \\ 0 & 1/3\,\mathbf{t}[n-2](Y) \end{pmatrix}}^{L+Q(\mathbf{t}[n-2],\cdot)} \mathbf{p}[n] + \overbrace{\begin{pmatrix} 1/4 & 0 \\ 0 & 1/3 \end{pmatrix}}^{L'} \mathbf{p}[n-1]\,.$$

Unlike $\mathbf{P}[n]$, the vector $\mathbf{p}[n]$ can be expressed in the form $A_n\mathbf{p}[n-1]$ for a suitable matrix A_n:

PROPOSITION 13. *Let $A_n := (L + Q(\mathbf{t}[n-2], \cdot))^* L'$. Then for every $n \geq |Q|^2|\Gamma| + 3$ the matrix A_n exists and $\mathbf{p}[n] = A_n\mathbf{p}[n-1]$.*

The key of our further analysis is to replace the matrix A_n by $A = \lim_{n\to\infty} A_n$. Since $A_n = (L + Q(\mathbf{t}[n-2], \cdot))^* L'$, we have

$$A := (L + Q(\mathbf{t}, \cdot))^* L'$$

where we define $\mathbf{t} = \lim_{n\to\infty} \mathbf{t}[n]$. (Observe that $\mathbf{t}(pXq) = [pXq]$.) It is not immediate from Proposition 13 that A exists, but it can be proved:

PROPOSITION 14. *The matrix A exists and its spectral radius ρ satisfies $\rho < 1$.*

Proposition 14 is the technical core of this paper. Its proof is quite involved and relies on Perron-Frobenius theory [2]. We give a proof sketch and a full proof in [5].

EXAMPLE 15 *The termination probabilities $\mathbf{t}$ can be computed as the least solution of a nonlinear equation system [8, 11]. Applied to our example we obtain $\mathbf{t}(X) = 2 - \sqrt{2} \approx 0.586$ and $\mathbf{t}(Y) = 1$. Basic computations yield the following matrix A whose spectral radius is $\rho = 1/2$.*

$$A = \begin{pmatrix} 1/(2+\sqrt{2}) & 1/(4+2\sqrt{2}) \\ 0 & 1/2 \end{pmatrix}$$

3.2 Approximating the Distribution and a Tail Bound

We can assume $p_0 X_0 \in \mathcal{H}$ in the following, because otherwise, by Lemma 8, we would have $\mathbf{P}[n](p_0 X_0) = 0$ for $n \geq |Q|^2|\Gamma| + 3$, removing any need for further analysis.
The following theorem suggests an efficient approximation algorithm.

THEOREM 16. *Let $n_\perp := |Q|^2|\Gamma| + 3$ and $\widehat{\mathbf{p}}[n] := \mathbf{p}[n]$ for $n < n_\perp$ and $\widehat{\mathbf{p}}[n_\perp + n] := A^n \mathbf{p}[n_\perp]$ for $n \geq 0$. Then $\mathbf{p}[n] \leq \widehat{\mathbf{p}}[n]$ holds for all $n \geq 1$. Moreover, there exists d with $0 < d \leq 1$ and*

$$d \cdot \widehat{\mathbf{p}}[n](p_0 X_0) \leq \mathbf{p}[n](p_0 X_0) \leq \widehat{\mathbf{p}}[n](p_0 X_0) \; .$$

The proposition shows that $\mathbf{p}[n](p_0 X_0)$ and the approximation $\widehat{\mathbf{p}}[n](p_0 X_0)$ differ at most by a constant factor. Given A, the matrix powers A^n can be computed by repeated squaring, which allows to compute this upper bound in time $\mathcal{O}\left((|Q| \cdot |\Gamma|)^3 \cdot \log n\right)$ in the Blum-Shub-Smale model. To compute $A = (L + Q(\mathbf{t}, \cdot))^* L'$ itself, we can compute the matrix star via the matrix inverse, as stated in the preliminaries. Computing the vector $\mathbf{t}$ of termination probabilities requires a more detailed discussion. The vector is the least solution of a nonlinear equation system, and its components may be irrational and even non-expressible by radicals [8, 11]. However, there are several ways to compute at least upper bounds on $\mathbf{t}$ (which suffices to obtain upper bounds on $\mathbf{p}[n]$, as A depends monotonically on $\mathbf{t}$), or lower-bound approximations sufficiently accurate for all practical purposes, see [5] for a discussion. Theorem 16 provides a tail bound for $\mathbf{p}[n](p_0 X_0)$:

COROLLARY 17. *We have $\mathbf{p}[n](p_0 X_0) \in \Theta\left(\rho^n\right)$.*

EXAMPLE 18 *Since in our example Proposition 13 holds already for $n \geq 2$, we have $\widehat{\mathbf{p}}[n] = A^{n-1}\mathbf{1}$ for $n \geq 1$. With the matrix A from Example 15 and using $\mathbf{p}[n] \leq \widehat{\mathbf{p}}[n]$ we obtain:*

$$\mathbf{p}[2] \leq 0.5 \cdot \mathbf{1}, \quad \mathbf{p}[5] \leq 0.07 \cdot \mathbf{1}, \quad \mathbf{p}[17] \leq 10^{-4} \cdot \mathbf{1}, \quad \mathbf{p}[65] \leq 10^{-19} \cdot \mathbf{1}, \quad \ldots$$

Binary search can be used to determine that $n = 18$ is the least number n for which $\mathbf{p}[n] \leq \widehat{\mathbf{p}}[n] \leq 10^{-5} \cdot \mathbf{1}$ holds, so the comparison with Example 6 shows that the overapproximation is quite tight here. As $\rho = 1/2$, Corollary 17 yields $\mathbf{p}[n](p_0 X_0) \in \Theta\left(1/2^n\right)$.

3.3 Approximating the Expectation

We define an approximation method for the expectation $EM_{p_0X_0}$, and bound its error. As mentioned below Proposition 7, we have $EM_{p_0X_0} = \sum_{n=1}^{\infty} \mathbf{p}[n](p_0X_0)$, which can be (under-) approximated by the partial sums $\sum_{n=1}^{k} \mathbf{p}[n](p_0X_0)$. The values $\mathbf{p}[n](p_0X_0)$ can be computed using Proposition 11.

The following theorem gives error bounds on this approximation method and shows that it converges linearly, i.e., the number of accurate bits (as defined in [14]) is a linear function of the number of iterations. (Recall for the following statement that for a vector $\mathbf{v} \in \mathbb{R}^{\mathcal{H}}$ its 1-norm $\|\mathbf{v}\|_1$ is defined as $\sum_{h\in\mathcal{H}} |\mathbf{v}(h)|$, and that for a matrix B its 1-norm $\|B\|_1$ is the maximal 1-norm of its columns.)

THEOREM 19. *Let $UM_{p_0X_0}(k) := \sum_{n=1}^{k} \mathbf{p}[n](p_0X_0)$. For all $k \geq |Q|^2|\Gamma| + 3$*

$$EM_{p_0X_0} - UM_{p_0X_0}(k) \leq \|A^*\|_1 \, \|\mathbf{p}[k]\|_1 \leq ab^k$$

where $a > 0$ and $0 < b < 1$ are computable rational numbers. Hence, the sequence $(UM_{p_0X_0}(k))_k$ converges linearly to $EM_{p_0X_0}$.

The computation procedure of the constants a and b from Theorem 19 is somewhat involved, but the first inequality of Theorem 19 gives concrete error bounds as well:
EXAMPLE 20 *Using Proposition 11 we compute $\sum_{n=1}^{12} \mathbf{p}[n](X) = 1.5731\ldots$ and furthermore $\|\mathbf{p}[12]\|_1 \approx 0.00042$. We have $\|A^*\|_1 = 1 + \sqrt{2} \approx 2.4$. Theorem 19 yields*

$$1.57 < EM_X \leq 1.5731\ldots + \|A^*\|_1 \cdot \|\mathbf{p}[12]\|_1 < 1.58\,.$$

4 Finiteness of the Expected Memory Consumption

In this section we study the complexity of the finite-expectation problem that asks whether the expectation of the memory consumption is finite.

4.1 Expected Memory Consumption of pPDA

For pPDA we can show the following theorem.

THEOREM 21. *The problem whether $EM_{p_0X_0}$ is finite is decidable in polynomial space.*

The proof is based on the following proposition which strengthens Proposition 14 from the previous section which stated that, under the assumption that $EM_{p_0X_0}$ is finite, the spectral radius ρ of A satisfies $\rho < 1$.

PROPOSITION 22. *Suppose $\mathcal{P}(M_{p_0X_0} < \infty) = 1$. Then the matrix A exists. Moreover, its spectral radius ρ satisfies $\rho < 1$ if and only if $EM_{p_0X_0}$ is finite.*

The condition $\mathcal{P}(M_{p_0X_0} < \infty) = 1$ can be checked in polynomial space [9]. If it does not hold, then clearly $EM_{p_0X_0} = \infty$. Otherwise one checks $\rho \geq 1$. Roughly speaking, this can be done in polynomial space because the matrix A is given in terms of the termination probabilities $\mathbf{t}$ which can be expressed in the existential theory of the reals.

We can also show that this upper complexity bound from Theorem 21 cannot be significantly lowered without a major breakthrough on long-standing and fundamental problems on numerical computations, namely the SQRT-SUM and the PosSLP problems [1, 11, 5]:

THEOREM 23. *The* PosSLP *problem is P-time many-one reducible to the decision problem whether the expected maximal height of a pPDA is finite.*

It follows that SQRT-SUM is (Turing) reducible to the finite-stack problem, because SQRT-SUM is (Turing) reducible to PosSLP [1, 11].

4.2 Expected Memory Consumption of pBPA

Now we show that for pBPA the finite-expectation problem can be decided in polynomial time. Let us fix a pBPA $\Delta = (\{p\}, \Gamma, \delta, Prob)$, and fix an initial configuration $X_0 \in \Gamma$. Let Γ_0 denote the set of all symbols $Y \in \Gamma$ such that $Y\Gamma^*$ is reachable from X_0. Let *Term* be the set of all symbols $X \in \Gamma_0$ such that $\mathbf{t}(X) = 1$, i.e., a run from a *Term*-symbol terminates almost surely. We define $NTerm = \Gamma_0 \setminus Term$. The following proposition follows from [11].

PROPOSITION 24. *The sets Term and NTerm can be computed in polynomial time.*

ALGORITHM DECIDING WHETHER EM_{X_0} IS FINITE:

1. Compute the sets *Term* and *NTerm* (using Proposition 24).
2. Decide in polynomial time [5] whether all $Y \in NTerm$ satisfy $\mathcal{P}(M_Y < \infty) = 1$. If no, then stop and return 'no'.
3. Decide in polynomial time [5] whether all $Y \in Term$ satisfy $EM_Y < \infty$. If no, then return 'no'. Otherwise return 'yes'.

THEOREM 25. *The above algorithm is polynomial. It returns 'yes' iff EM_{X_0} is finite.*

5 Conclusions

We have investigated the memory consumption of probabilistic pushdown automata (pPDA). Technically speaking, we have studied the random variable M returning the maximal stack height of a pPDA. In [9] a PSPACE algorithm was provided for deciding whether the runs with $M = \infty$ have nonzero probability, but the distribution of M and its expectation have not been studied. For computing the distribution of M, we have shown that the exponential blow-up of the naive method can be avoided using a system of linear equations. We have also provided an approximation method that gives upper bounds. This can be used, e.g., for providing space that suffices with a probability of, say, 99%.

Computing the expectation EM was mentioned in [9] as "harder problem" and left open. Using novel proof techniques, we have provided a rather complete solution. We have shown that whether the expected maximal stack height of a pBPA is finite can be decided in polynomial time, while for general pPDA the problem is in PSPACE. By means of a reduction to the PosSLP and SQRT-SUM problems we have furthermore shown that this complexity cannot be significantly lowered without major breakthroughs. Finally, we have defined an iterative method for approximating the expected maximal stack height, and

have shown that it converges linearly. The complexity of the decision problem $EM_{p_0X_0} < k$ for a finite bound k is an open question.

References

[1] E. Allender, P. Bürgisser, J. Kjeldgaard-Pedersen, and P. B. Miltersen. On the complexity of numerical analysis. In *IEEE Conference on Computational Complexity*, pages 331–339. IEEE Computer Society, 2006.

[2] A. Berman and R.J. Plemmons. *Nonnegative matrices in the mathematical sciences*. Academic Press, 1979.

[3] R.D. Blumofe and C.E. Leiserson. Scheduling multithreaded computations by work stealing. *Journal of the ACM*, 46(5):720–748, 1999.

[4] A. Bouajjani and J. Esparza. Rewriting models of boolean programs. In *Proceedings of RTA 2006*, Seattle, USA, 2006.

[5] T. Brázdil, J. Esparza, and S. Kiefer. On the memory consumption of probabilistic pushdown automata. Technical Report FIMU-RS-2009-07, Masaryk University, 2009. Available at `http://www.fi.muni.cz/reports/files/2009/FIMU-RS-2009-07.pdf`.

[6] T. Brázdil, J. Esparza, and A. Kučera. Analysis and prediction of the long-run behavior of probabilistic sequential programs with recursion. In *FOCS'05*, pages 521–530, 2005.

[7] J. Esparza and K. Etessami. Verifying probabilistic procedural programs. In *FSTTCS 2004*, pages 16–31, 2004.

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

[9] J. Esparza, A. Kučera, and R. Mayr. Quantitative analysis of probabilistic pushdown automata: Expectations and variances. In *LICS'05*, pages 117–126. IEEE, 2005.

[10] K. Etessami and M. Yannakakis. Recursive concurrent stochastic games. *Logical Methods in Computer Science*, 4(4), 2008.

[11] K. Etessami and M. Yannakakis. Recursive markov chains, stochastic grammars, and monotone systems of nonlinear equations. *Journal of the ACM*, 56(1):1–66, 2009.

[12] E.M. Hahn, H. Hermanns, B. Wachter, and L. Zhang. INFAMY: An infinite-state Markov model checker. In *CAV*, LNCS 5643, pages 641–647, 2009.

[13] R.A. Horn and C.A. Johnson. *Matrix Analysis*. Cambridge University Press, 1985.

[14] S. Kiefer, M. Luttenberger, and J. Esparza. On the convergence of Newton's method for monotone systems of polynomial equations. In *STOC 2007*, pages 217–226, 2007.

[15] G.J. Narlikar and G.E. Belloch. Space-efficient scheduling of nested parallelism. *ACM TOPLAS*, 21(1):138–173, 1999.

[16] A. Remke, B. Haverkort, and L. Cloth. CSL model checking algorithms for QBDs. *Theoretical Computer Science*, 382(1):24–41, 2007.

[17] P. Schnoebelen. The verification of probabilistic lossy channel systems. In *Validation of Stochastic Systems*, LNCS 2925, pages 445–465. Springer, 2004.

[18] M. Yannakakis and K. Etessami. Checking LTL properties of recursive Markov chains. In *QEST 2005*, pages 155–165, 2005.